TESIS - KI2501

Peningkatan Kinerja *Greedy Perimeter Stateless Routing* (GPSR) Berbasis *Overlay Network* pada VANET

FAISHAL HALIM S

NRP. 5115201040

DOSEN PEMBIMBING:

Dr.Eng. Radityo Anggoro, S. Kom., M. Sc

Prof.Ir. Supeno Djanali, M. Sc, Ph. D

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN
DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

THESIS - KI2501

Improvement Performance of Greedy Perimeter Stateless Routing (GPSR) Based on Overlay Network in VANET

FAISHAL HALIM S

NRP. 5115201040

SUPERVISOR:

Dr.Eng. Radityo Anggoro, S. Kom., M. Sc

Prof.Ir. Supeno Djanali, M. Sc, Ph. D

MAGISTER PROGRAMME
NET CENTRIC COMPUTING
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND COMMUNICATION
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2018

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya.

Oleh:
Faishal Halim S
NRP. 5115201040

Dengan judul:
Peningkatan Kinerja *Greedy Perimeter Stateless Routing* (GPSR) Berbasis *Overlay Network* pada VANET

Tanggal Ujian: 4 Januari 2018
Periode Wisuda: Maret 2018

Disetujui Oleh:

Dr.Eng. Radityo Anggoro, S. Kom., M. Sc (Pembimbing 1)
NIP. 19841016200812 1 002

Prof. Ir. Supeno Djanali, M.Sc, Ph.D. (Pembimbing 2)
NIP. 19480619197301 1 001

Tohari Ahmad, S.Kom, MIT, Ph.D. (Penguji 1)
NIP. 19750525200312 1 002

Royyana Muslim I, S.Kom, M.Kom, Ph.D. (Penguji 2)
NIP. 19770824200604 1 001

Waskitho Wibisono, S.Kom, M.Eng, Ph.D. (Penguji 3)
NIP. 19741022200003 1 001

Dekan FTIK – ITS

Dr. Agus Zainal Arifin, S.Kom, M.Kom.
NIP. 197208091995121001

[Halaman ini sengaja dikosongkan]

PENINGKATAN KINERJA *GREEDY PERIMETER STATELESS ROUTING* (GPSR) BERBASIS *OVERLAY NETWORK* PADA VANET

Nama Mahasiswa	: Faishal Halim S
NRP	: 5115201040
Pembimbing	: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc
Co-Pembimbing	: Prof. Ir. Supeno Djanali, M.Sc, Ph.D.

ABSTRAK

VANET menjadi teknologi yang menarik dalam hal keselamatan berkendara dan layanan jaringan yang nyaman bagi pengguna kendaraan. Namun, mobilitas tinggi pada VANET membuat topologi sering berubah mengakibatkan tidak tersedianya rute dan kegagalan komunikasi. GPSR merupakan protokol *routing* geografis yang bekerja menggunakan informasi lokasi *node* tetangga dan jangkauan transmisi ke lokasi *node* tujuan terdekat. Namun, GPSR tidak selalu menemukan rute yang optimal, karena tidak semua *node* yang terdekat dapat meneruskan paket menuju tujuan khususnya skenario lalu lintas padat dan banyak persimpangan.

Pada penelitian ini kami mengusulkan *route discovery protocol Dynamic Source Routing* (DSR) untuk memastikan rute yang akan dikirimkan paket data memiliki *availability* yang tinggi. Selain itu kami juga mengusulkan konsep *overlay network* dengan menyimpan titik lokasi persimpangan yang dilalui pada saat proses *route discovery* sehingga proses pengiriman data nantinya dapat diandalkan dengan tidak terpaku pada *intermediate node* namun mengacu pada lokasi titik yang disimpan. Untuk menguji usulan kami dengan membandingkan performa GPSR modifikasi dan GPSR konvensional dengan metrik seperti *packet delivery ratio, end-to-end delay* dan *routing overhead* pada skenario peta grid dan real dengan variasi jumlah *node* dan kecepatan *node* yang berbeda menggunakan simulator NS2. Hasil simulasi menunjukan pendekatan yang kami usulkan meningkatkan *packet delivery ratio* 73.20% dan mereduksi 387 paket *routing overhead* pada kecepatan 10 m/s.

Kata kunci: DSR, GPSR, *OVERLAY, NETWORK*, VANET.

[Halaman ini sengaja dikosongkan]

IMPROVEMENT PERFORMANCE OF GREEDY PERIMETER STATELESS ROUTING (GPSR) BASED ON OVERLAY NETWORK IN VANET

Name	*: Faishal Halim S*
NRP	*: 5115201040*
Supervisor	*: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc*
Co-Supervisor	*: Prof. Ir. Supeno Djanali, M.Sc, Ph.D.*

ABSTRACT

VANET becomes an exciting technology in terms of driving safety and convenient network services for drivers. However, high mobility in VANET makes topology often change and resulting in unavailability of routes and communication failures. GPSR is a geographic routing protocol that works using the neighbour node location information and transmission range to the location of the closest destination node. However, GPSR does not always find an optimal route, not all nearby nodes can forward packets to destinations, especially heavy traffic scenarios and many intersections.

In this study, we propose a route discovery protocol Dynamic Source Routing (DSR) to ensure the route to send data packets have high availability. In addition, we also proposed the concept of overlay network by storing location where the intersection is passed during the route discovery process, data transmission process can be rendered by not referring to intermediate node but refers to the location of the stored point. To analyse our proposals by comparing the performance of conventional GPSR modifications and GPSR such as packet delivery ratio, end-to-end delay and routing overhead in different scenarios by varying the node and speed using the NS2 simulator. The simulation results show that our proposed approach increases the packet delivery ratio by 73.20% and reduces 387 routing overhead packets at speed 10 m/s.

Keywords*:* DSR, GPSR, *OVERLAY, NETWORK,* VANET.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Bismillahirrohmanirohim.

Alhamdulillahirabil'alamin, segala puji bagi Allah Subhanahu Wata'alla, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tesis yang berjudul "**Peningkatan Kinerja *Greedy Perimeter Stateless Routing* (GPSR) Berbasis *Overlay Network* pada VANET**" dengan baik dan tepat waktu.

Dalam pelaksanaan dan pembuatan Tesis ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Kedua orang tua penulis, yang telah memberikan dukungan moral, spiritual dan material, semangat, perhatian, selalu setia dan sabar dalam menghadapi curhatan dari penulis, serta kakak dan adik yang selalu memberikan doa yang tiada habisnya untuk penulis.
3. Bapak Dr.Eng Radityo Anggoro, S.Kom, M.Sc., dan Bapak Prof. Supeno Djanali, M.Sc, Ph.D. selaku dosen pembimbing pertama dan kedua, saya ucapkan banyak terimakasih atas bimbingan, arahan, serta bantuan ide dan masukan untuk menyelesaikan Tesis ini.
4. Bapak Waskitho Wibisono, S. Kom., M.Eng., Ph.D, Dr.Eng. selaku ketua program studi pasca sarjana jurusan Teknik Informatika ITS, Ibu Bilqis Amalia., S.Kom., selaku dosen wali penulis, Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom selaku sekretaris prodi S2 dan dosen wali pengganti.
5. Bapak Royyana Muslim Ijtihadie S. Kom, M. Kom, Ph.D selaku atasan dan senior di kantor maupun kelas yang selalu memberi masukan dan informasi mengenai ilmu networking kepada penulis.
6. Mba Lina, Mba Yasna, Pak Yudi, Pak Sugeng, Mas Kunto, Mas Edi Lukito dan segenap Dosen dan Karyawan Tenik Informatika yang telah memberikan segala ilmu, bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.
7. Bunda Nur Sofi Farida yang selalu membantu dan memberikan semangat, masukan kepada penulis mengenai penampilan, penulisan buku, kelengkapan yudisium dan masih banyak lagi, saya ucapkan terimakasih sebesar-besarnya.
8. Mbak Mala dan Mas Pradhika staff UPT Bahasa yang selalu saya repoti pada saat tes Bahasa inggris saya ucapkan terima kasih banyak sehingga penulis dapat melengkapi berkas yudisiumnya.
9. Grezio, Erna, Santi, Rahmi, Ihsan dan Bagus Gede sebagai teman yang selalu memotivasi dan membantu dalam menyelesaikan Tesis ini.
10. Seluruh mahasiswa pasca sarjana 2015 dan 2016, selama bersama kalian sadar atau tidak telah membentuk karakter dan kepribadian penulis.
11. Seluruh staff Tendik maupun Dosen ITS khususnya di lingkungkan DPTSI/PUSKOM ITS yaitu SubDirektorat IKTI, Pusat Pelayanan dan Pusat Pengembangan yang tidak bisa sebutkan satu-persatu.

12. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tesis ini.

Kesempurnaan tentu sangat jauh tercapai pada Tesis ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Januari 2018

Faishal Halim S

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR LAMPIRAN

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Semakin pesatnya perkembangan teknologi, aspek komunikasi antar komputer menjadi sangat penting. Salah satu cara komputer berkomunikasi adalah menggunakan jaringan *ad-hoc*. Dalam jaringan *ad-hoc* setiap komputer terkoneksi satu sama lain secara langsung tanpa bergantung pada *router* atau *access point*. Salah satu system yang menggunakan jaringan *ad-hoc* adalah VANET.

VANET (*Vehicular ad-hoc Network*) sebuah teknologi baru yang digunakan untuk komunikasi antar kendaraan (*inter-vehicle*) dengan penambahan perangkat. VANET merupakan bagian penting dalam penerapan ITS (*Intelligent Transportation System*) (Andrisano dkk, 2000), VANET dapat digunakan antara lain untuk mengurangi resiko kecelakaaan (Batool & Khan, 2005), mengatur trafik lalu lintas kendaraan yang ada di perempatan jalan (Biswas, 2006).

VANET merupakan pengembangan dari *Mobile Ad Hoc Network* (MANET) yang diaplikasikan dalam kendaraan. Jaringan VANETs memungkinkan kendaraan saling berkomunikasi tanpa membutuhkan pengaturan infrastruktur tersentral ataupun *server* yang digunakan untuk mengontrol kerjanya (Anggoro dkk, 2003) (Hussein dkk, 2005). VANETs memiliki banyak *routing protocol*, salah satunya adalah *Greedy Perimeter Stateless Routing* (GPSR) yang akan digunakan pada penelitian ini.

Greedy Perimeter Stateless Protocol (GPSR) menggunakan informasi *geographic* untuk mengetahui posisi setiap *node* sehingga dapat memilih *node* tetangga (*intermediate*) terdekat dengan *node* tujuan sebagai perantara pengiriman paket (Karp & Kung, 2000). Mekanisme *forwarding* paket pada GPSR selalu mencari *node* mana yang paling dekat dengan *destination* secara *greedy* tanpa tahu apakah *node* tersebut benar-benar bisa digunakan untuk mengirimkan paket menuju *node* tujuan. Hal ini sangat rawan terjadi kegagalan pemilihan *intermediate node* pada kondisi jalan perkotaan yang ramai dan banyak persimpangan yang dapat mengakibatkan proses *greedy* pada protokol *routing* GPSR gagal.

Protokol *reactive Dynamic Source Routing* (DSR) menawarkan sebuah pendekatan *route discovery* dengan membanjiri pesan *route request* secara dinamis pada sebuah jaringan hingga mencapai *node* tujuan, kemudian merespon pesan *route reply* dan membawa informasi rute menuju *node* pengirim (Kumar & Routray, 2017). *Route discovery* ini dapat memastikan jalur *node* yang dilewatinya tersedia menuju tujuan dengan menyimpan semua *node* yang dilewatinya sehingga dapat menambah informasi pada saat proses *greedy.* Oleh karena itu, performa GPSR perlu ditingkatkan dengan cara menambahkan metode *route discovery* DSR yang akan digunakan untuk mencari *availability* rute menuju *destination* sebelum mengirimkan data.

Namun, Karena karakteristik VANET yang membuat setiap *node* dapat bergerak secara cepat (Djenouri et al., 2008), *routing* GPSR dengan mengadopsi *route discovery* DSR masih memiliki kelemahan apabila *intermediate node* yang sudah disimpan bergerak berpindah area lain. Oleh karena itu, peneliti juga mengusulkan diterapkannya *overlay network* dengan memanfaatkan proses *route discovery* DSR untuk menyimpan letak posisi tiap persimpangan yang dilewati sehingga pada saat pengiriman data dapat dimodifikasi dengan merubah *node* tujuan menjadi setiap lokasi persimpangan yang ada sebelum mencapai *node* tujuan. Sehingga *header* pada *node* pengirim fokus pada titik lokasi yang telah ditandai dan tidak mempermasalahkan *intermediate node* yang berubah-ubah.

Pada penelitian ini, penulis menggunakan istilah "GPSR modifikasi" untuk metode yang diusulkan dengan memodifikasi *Greedy Perimeter Stateless Routing* (GPSR) dengan mengadopsi *route discovery* protokol DSR dan menerapkan konsep *overlay network*. Penulis melakukan uji coba dan membandingkan performa protokol GPSR dan GPSR modifikasi dengan beberapa variasi *node* dan kecepatan.

1.2 Perumusan Masalah

Berdasarkan masalah yang diuraikan pada latar belakang maka rumusan masalah yang akan diselesaikan adalah sebagai berikut:

1. Bagaimana cara mengatasi kegagalan *greedy forwarding* dalam meneruskan paket menuju *node* tujuan pada *routing* GPSR?
2. Bagaimana cara menerapkan *route discovery* DSR pada GPSR?
3. Bagaimana pengaruh konsep *overlay network* terhadap performa GPSR?

1.3 Tujuan Penelitian

Tujuan penelitian ini adalah untuk:

1. Mengetahui pengaruh *route discovery* DSR dan konsep *overlay network* terhadap performa *routing protocol* yang telah dimodifikasi.
2. Meningkatkan performa *routing* GPSR diukur dengan *route metric* (*packet delivery ratio, routing overhead* dan *end to end delay*).

1.4 Manfaat penelitian

Manfaat dari penelitian ini adalah sebagai pedoman yang dapat digunakan untuk mengukur kinerja dari *routing* protokol berbasis geografis *Greedy Perimeter Stateless Routing* (GPSR) di lingkungan VANET.

1.5 Kontribusi

Kontribusi penelitian ini adalah untuk meningkatkan jumlah paket yang berhasil terkirim dengan mempertimbangkan faktor *availability* rute pengiriman pada protokol *routing* GPSR yang telah dimodifikasi.

1.6 Batasan Penelitian

Implementasi VANET pada dunia nyata membutuhkan biaya yang tidak sedikit, maka dalam penelitian ini dibatasi sebagai berikut:

1. *Routing* protokol yang digunakan adalah GPSR.
2. Implementasi dan ujicoba menggunakan simulator NS-2.
3. *Node* pengirim dan *node* tujuan tetap.

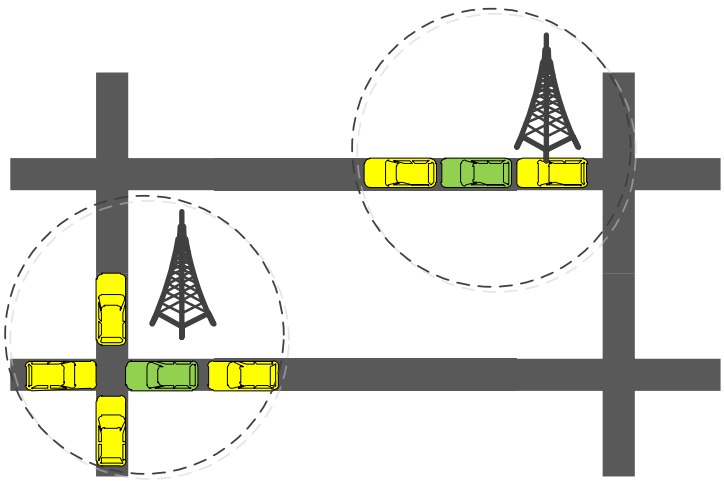
[halaman ini sengaja dikosongkan]

BAB 2
DASAR TEORI DAN KAJIAN PUSTAKA

2.1 *Vehicular Ad Hoc Network* (VANET)

VANET (*Vehicular Ad Hoc Network*) adalah pengembangan dari MANET (*Mobile Ad Hoc Network*). VANET berjalan menggunakan jaringan *wireless* berbasis *Ad Hoc* yang tambahkan pada kendaraan bergerak (mobil, kendaraan umum, dll). VANET menjadi bagian dari ITS (*Intelligent Transportation System*), pada beberapa tahun terakir banyak penelitian yang dilakukan pada VANET. Secara umum VANET dikembangkan untuk membantu kendaraan-kendaraan untuk berkomunikasi dan memelihara jaringan komunikasi diantara kendaraan tanpa menggunakan bantuan dari *central base station* atau *controller* (Rehman et al., 2013).

Kendaraan pada VANET dapat berkomunikasi dengan kendaraan lain komunikasi ini dapat disebut dengan komunikasi V2V (*Vehicle to Vehicle*), selain komunikasi V2V kendaraan dapat berkomunikasi dengan infrastruktur seperti RSU (*Road Side Unit*) komunikasi ini disebut dengan V2I (*Vehicle to Infrastructure*) atau komunikasi V2R (*Vehicle to Roadside*), komunikasi V2I dan V2R dapat dilihat pada Gambar 2..



Gambar 2.1 *Overall framework of* VANET (Rehman et al. 2013)

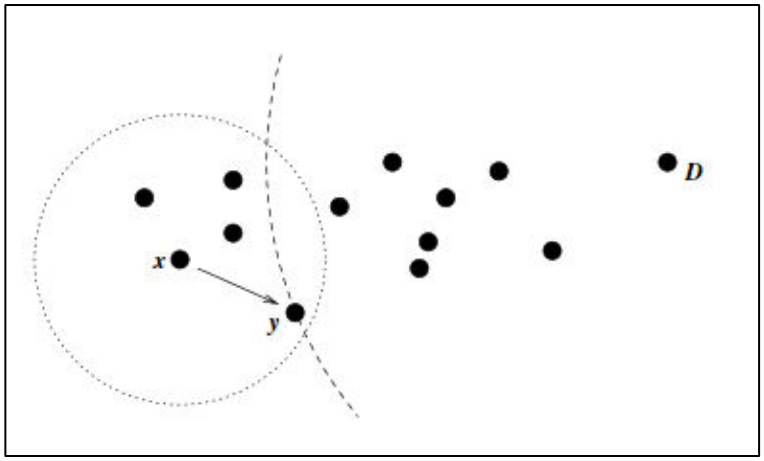
2.2 *Greedy Perimeter Stateless Routing* (GPSR)

Greedy Perimeter Stateless Routing merupakan *Geographical Routing Protocol* yang menggunakan *position-based routing*, dimana setiap *node* saling mengetahui posisi sendiri dan posisi tetangga terdekatnya. *Routing* protokol GPSR akan mengirimkan *hello message* secara berkala untuk memperbarui informasi lokasi geografis *node* yang msaih berada pada jangkauan transmisi (Silmi, 2016).

Protokol GPSR menggunakan dua metode untuk melakukan pengiriman paket data, yaitu (Karp & Kung, 2000):

1. *Greedy forwarding*

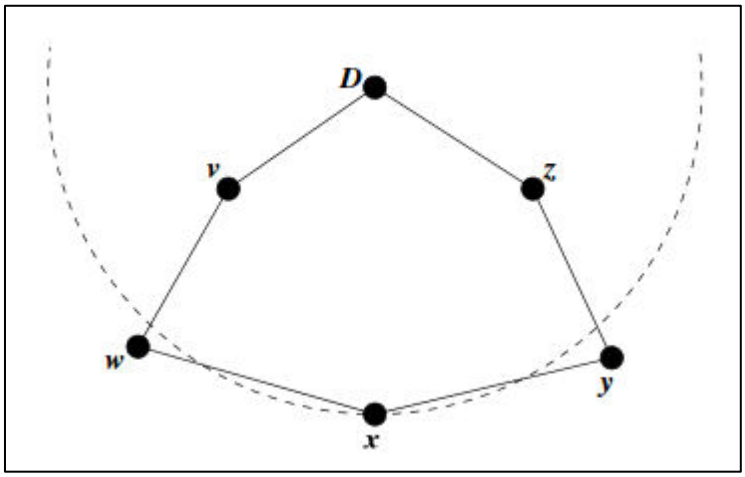
   Greedy forwarding merupakan metode utama yang digunakan untuk pengiriman paket data. *Greedy forwarding* akan meneruskan paket data kepada *node* yang paling dekat dengan *destination node*. Gambaran tentang metode *greedy forwarding* dapat dilihat pada Gambar2.2 *node x* sebagai *node* pengirim menunjuk *node y* sebagai perantara pengirim paket karena jarak *node x* yang paling dekat dengan *node D* atau *node* tujuan.



Gambar 2.2 Metode *greedy forwarding* (Karp & Kung, 2000)

2. *Perimeter forwarding*

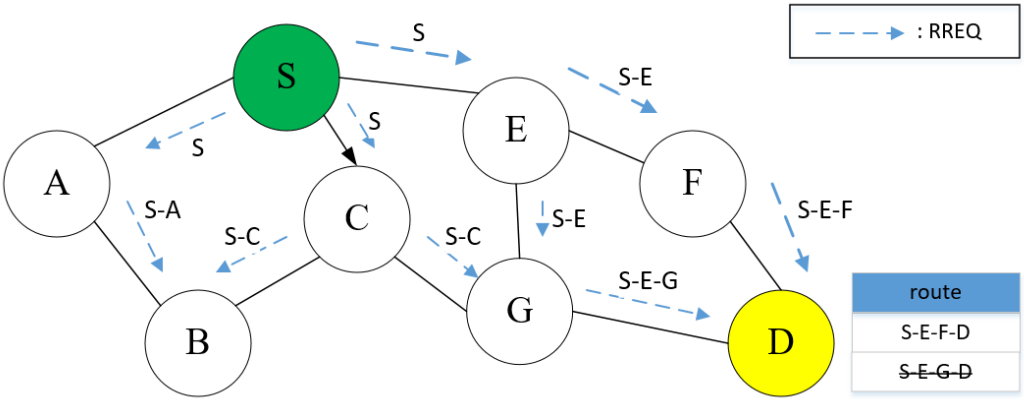
   Perimeter forwarding akan bekerja apabila *greedy forwarding* tidak bisa menemukan *node* yang paling dekat dengan *node* tujuan. Sehingga *perimeter forwarding* akan memanfaatkan *node* disekitarnya yang dapat mencapai *node* tujuan, namun jalur yang dilewatinya lebih panjang karena harus melewati banyak *hop* (Shelly & Babu, 2015). Gambaran tentang *perimeter forwarding* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Metode *perimeter forwarding* (Karp & Kung, 2000)

2.3 *Dynamic Source Routing* (DSR)

Dynamic Source Routing merupakan *routing* protokol yang bersifat *reactive* yang artinya protokol ini akan melakukan *route discovery* jika sebuah *node* inisiator ingin berkomunikasi dengan *node* yang lain, oleh karena itu *network traffic* dapat diturunkan. Sebelum melakukan pengiriman paket data, DSR akan melakukan *route discovery* kemudian menyimpan *route path* pada *header* paket yang digunakan untuk pengiriman paket data. Pada *routing* DSR, *intermediate node* tidak perlu mengurus informasi *routing*. Proses *route discovery* dimulai saat sebuah *node source* ingin berkomunikasi dengan *node destination*. *Node source* akan mengirim *route request* (RREQ) menuju *node destination*. *Node intermediate* akan menyisipkan dirinya pada *route path* yang ada pada *header* RREQ jika belum pernah menerima RREQ tersebut dan menyimpan nomor RREQ pada *route cache,* lalu *node* tersebut akan melakukan *broadcast* ulang ke *node* yang lain. Apabila pada *route cache* sudah ada nomor RREQ tersebut maka paket RREQ akan di *drop* (Patel, 2014). Mekanisme *route discovery* pada DSR dapat dilihat pada Gambar 2.4 sampai Gambar 2.6

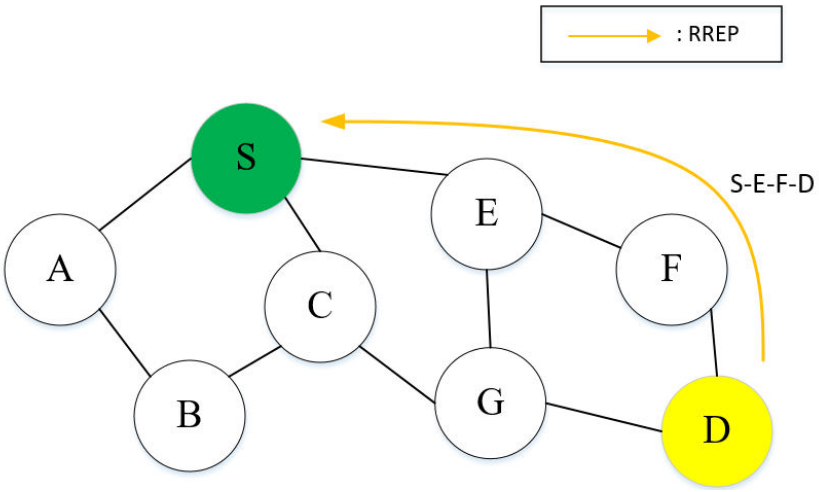


Gambar 2.4 Mekanisme *broadcast route request* (RREQ)

Sebelum mengirimkan paket data dari *node* S (*node* pengirim) menuju *node* D (*node* tujuan), *node* pengirim melakukan *route request (*RREQ*)* dengan cara membroadcast ke semua *node* tetangga yang dapat dijangkau seperti pada gambar diatas *node* S mem*broadcast* menuju *node* A, C dan E. Kemudian *node* A, C dan E melanjutkan RREQ kepada *node* tetangganya masing-masing hingga mencapai *node* tujuan.

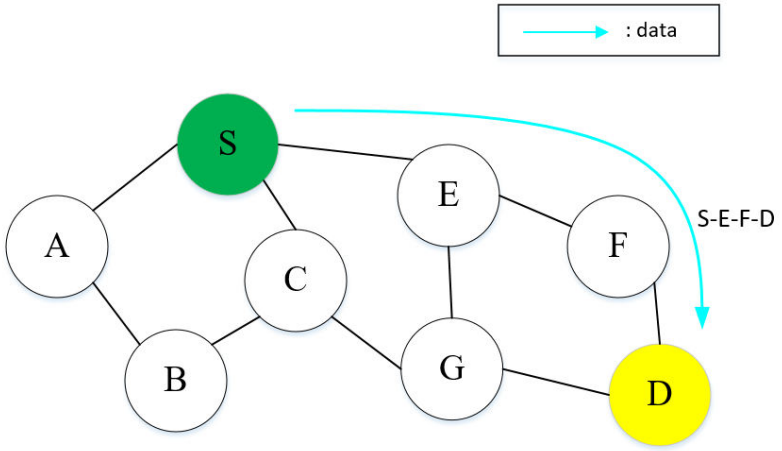
Setiap *intermediate node* akan meneruskan *broadcast route request (*RREQ*)* menuju *node* tetangga yang berada didekatnya. Apabila suatu *node* menerima pesan *route request* yang sama dengan id sebelumnya maka paket akan di *drop*.

Pada Gambar 2.4 *Node* tujuan mendapatkan dua paket RREQ, namun *route discovery* pada *routing* DSR hanya merespon paket RREQ yang diterima paling cepat, karena dengan asumsi pesan RREQ yang paling cepat mempunyai ketersediaan *node* yang lebih baik atau rute yang lebih pendek. Sehingga pada tabel *route* Gambar 2.4 jalur S-E-G-D akan di *drop* karena *node* tujuan telah menerima pesan RREQ yang lebih cepat melalui jalur S-E-F-D



Gambar 2.5 Mekanisme *route reply* (RREP) *destination node*

Setelah *destinination node* mendapatkan pesan RREQ kemudian merespon pesan dengan mengirimkan pesan *route reply* (RREP) menuju *node* tujuan secara *unicast* dengan melewati jalur *node-node* yang sudah ditandai pada saat melakukan *route request* (RREQ).

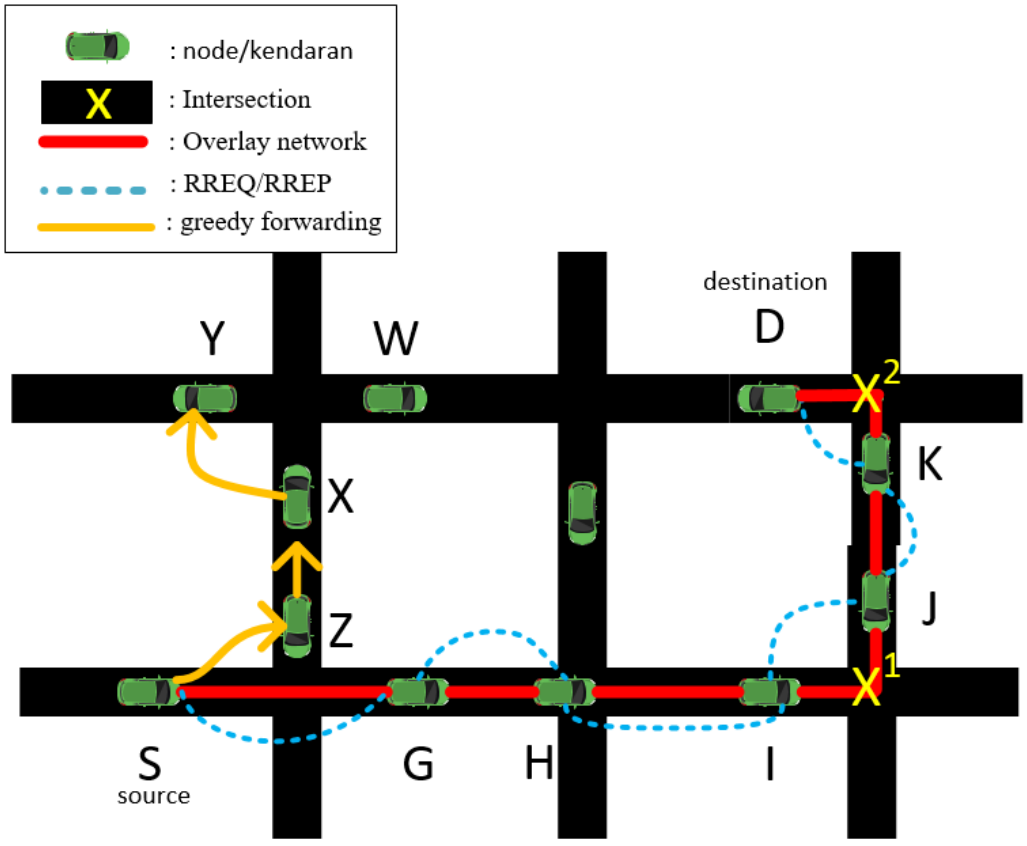


Gambar 2.6 Mekanisme pengiriman data

Sehingga *source node* akan mengetahui *node-node* yang mempunyai ketersediaan tinggi untuk mendistribusikan paket data dengan mengambil jalur yang telah dilalui pada saat proses *route discovery* yaitu S – E – F - D.

2.4 *Overlay Network*

Overlay Network adalah suatu jaringan yang dibangun di atas jaringan lain, yang artinya setiap *node* saling terhubung melalui jaringan *virtual* diatas jaringan fisik (Galan & Gazo, 2011). *Node* atau simpul pada *overlay network* dapat terhubung secara *virtual* atau logis melalui banyak jalur fisik yang dilewatinya. Biasanya *overlay network* berjalan pada TCP/IP *stack* dengan memanfaatkan lapisan yang mendasarinya (Firdhous, 2011). Pada penelitian ini konsep *overlay network* digunakan pada proses pengiriman paket data. *Node* sumber secara langsung (*direct*) mengirimkan paket data menuju titik lokasi yang sudah disimpan atau *node destination* menggunakan *overlay network*, namun jika dilihat dari *physical network* pengiriman paket data harus melalui banyak *node*.



Gambar 2.7 Konsep *overlay network*

Tabel 2.1 Perbandingan *link overlay* dan *physical path*

Link Overlay	*Physical Path*
$S - X^1 - X^2 - D$	S – G – H – I – J – K – D

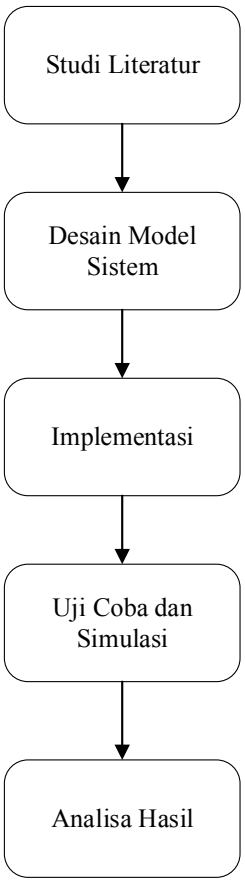
Pada gambar 2.7 terdapat perbedaan jalur yang dilewati antara metode GPSR dengan GPSR menggunakan *overlay network*. Algoritma GPSR akan mencari *node* terdekat lalu mem*forward* data tanpa tahu *node* tersebut dapat mencapai *node* tujuan yang rawan mengakibatkan kegagalan komunikasi atau paket di *drop* seperti contoh pada gambar 2.7 jalur yang akan dipilih yaitu S – Z – X – Y. *Node* Y berjalan menjauh dari *node destination* sehingga komunikasi gagal terbentuk.

Dengan proses *route discover* DSR, *node* sumber akan mengirimkan RREQ secara *broadcast* untuk mencari *node* tujuan kemudian *node* tujuan akan merespon dengan mengirim RREP, sehingga jalur yang *available* menuju *node* tujuan terbentuk. Hasil dari *route discovery* DSR, *node* S (*source node*) secara *physical* mengirimkan data menuju *node* D (*destination node*) melalui *intermediate node* yaitu S – G – H – I – J – K –D. Namun kondisi VANET yang membuat kendaraan atau *node* terus bergerak rawan akan terjadinya *neighbor loss,* dengan konsep *overlay network* pada saat proses *node* tujuan merespon RREP akan menyimpan titik lokasi persimpangan yang dilaluinya sehingga pada saat mengirimkan data akan menjadi acuan tiap *node* menuju tujuan dengan memodifikasi *header routing* protokol GPSR. Sehingga pada Gambar 2.7 rute pengiriman yang terbentuk setelah menggunakan *overlay* adalah S – X^1 – X^2 – D.

[halaman ini sengaja dikosongkan]

BAB 3
METODA PENELITIAN

Metode penelitian ini melalui beberapa tahap meliputi (1) Studi Literatur, (2) Desain Model Sistem, (3) Implementasi, (4) Uji Coba dan simulasi, (5) Analisa Hasil. Alur tahapan-tahapan tersebut dapat dilihat pada Gambar 3. 1.



Gambar 3. 1 Alur metoda penelitian

3.1 Studi Literatur

Studi literatur merupakan ujung tombak berhasil tidaknya sebuah penelitian, Studi literatur dilakukan untuk menggali informasi dan menganalisa perkembangan metodologi yang berkaitan dengan penelitian ini. Beberapa referensi yang dibutuhkan berkaitan dengan penelitian ini adalah:

1. ITS *(Intelligent Transportation System)* dan VANET *(Vehicular ad-hoc Network).*

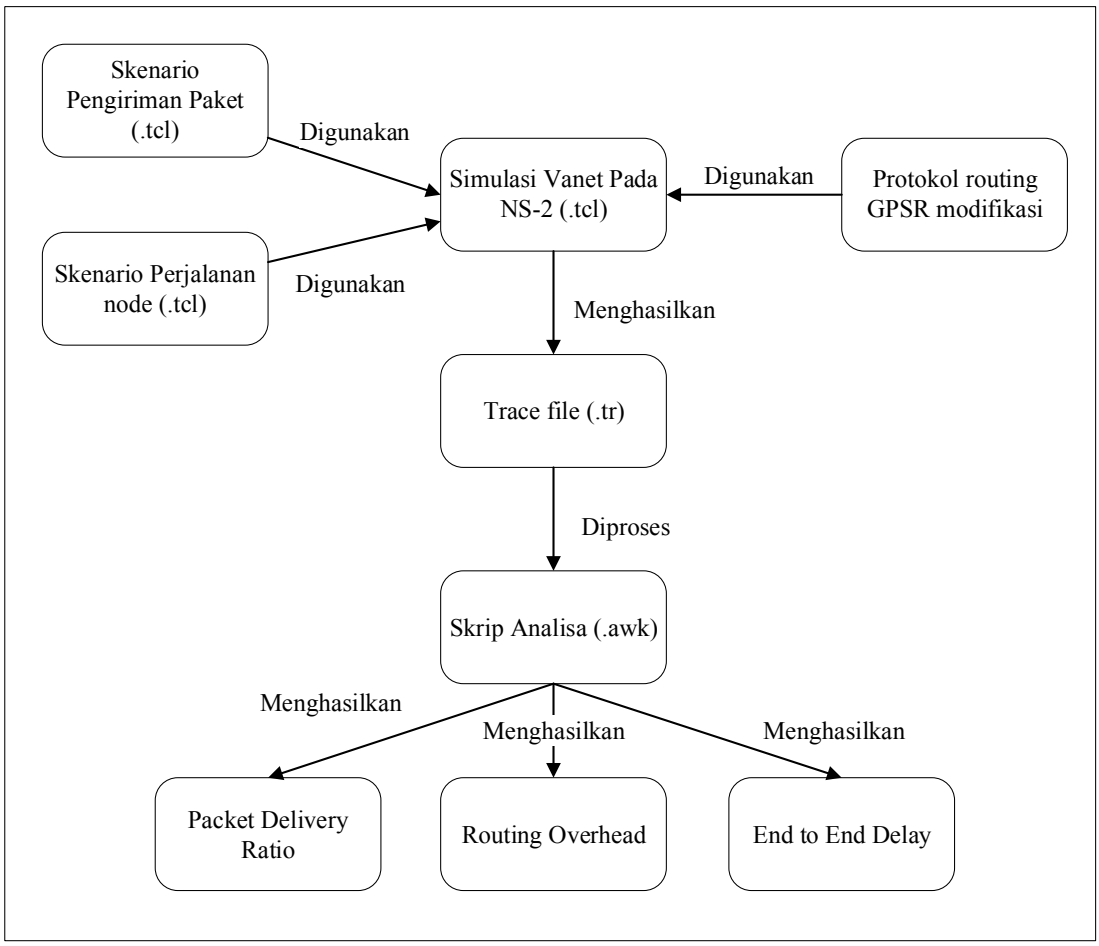
2. SUMO (*Simulation of Urban Mobility*) dan JOSM (*Java OpenStreetMap Editor*).
3. *Routing* protokol GPSR.
4. *Route discovery* protokol DSR.
5. *Overlay network.*

Tabel 3.1 Perbandingan protokol

Protokol *routing*	Kelebihan	Kekurangan
GPSR (Karp & Kung, 2000)	- Mencari *node* tetangga yang paling dekat dengan tujuan (*greedy perimeter*). - Jika tidak ada rute yang paling dekat maka rute yang dipilih menggunakan *node* di sekitar *node* sumber yang dapat menjangkau *node* tujuan (*forwarding perimeter)*.	- *Node* tetangga yang paling dekat belum tentu optimal karena tidak mempertimbangkan ketersediaan *next-hop* untuk meneruskan paket. - *Forwarding perimeter* menggunakan jalur yang lebih panjang (*right-hand rule*).
GPSR *Routing Strategy* (Hu et al., 2012)	- *Store-to-forward.* - Modifikasi *hello packet* untuk menandai kecepatan, arah dan kepadatan.	- Metode *store-to-forward* membutuhkan waktu lebih yang berakibat rawan *delay*. - Pemilihan *next-hop* berdasarkan kecepatan *node* yang < 10m/s kurang efektif apabila diterapkan pada VANET yang sesungguhnya.
P-GPSR (Dahmane & Lorenz, 2016)	- Memilih *relay node* dengan parameter kualitas link, stabilitas link dan kecepatan *node*.	- Hasil PDR dan *delay* masih kurang baik dibandingkan metode sebelumnya yaitu GPSR-L (Rao et al., 2008).

GPSR+PRedict (Houssaini et al., 2016)	- Menentukan *relay node* dengan parameter arah, kecepatan dan lokasi *node* sekarang dan mendatang.	- Kurang tepat jika diterapkan ke dunia nyata karena karakteristik vanet yang membuat *node* berubah-ubah.
GPSR-2P (Zaimi et al., 2016)	- Memilih dua *next-hop node* terdekat dengan tujuan sebagai *relay node*.	- Rawan terjadi duplikasi data yang terkirim. - Membutuhkan waktu proses karena setiap *node* inisiator akan mempertimbangkan dua *node next-hop.*
GPSR modifikasi (usulan)	- Melakukan *route discovery routing* DSR sebelum mengirimkan paket data untuk memastikan *route availibility.* - Menyimpan posisi setiap persimpangan yang dilewati pada saat proses RREP untuk mengaplikasikan *overlay network.* - Tidak mempermasalahkan *intermediate node* yang berubah-ubah.	- Adanya proses *route discovery* sehingga kemungkinan ada *delay* pada saat mengirimkan data. - Ukuran *header* yang bertambah karena modifikasi yang diusulkan.

3.2 Desain Model Sistem

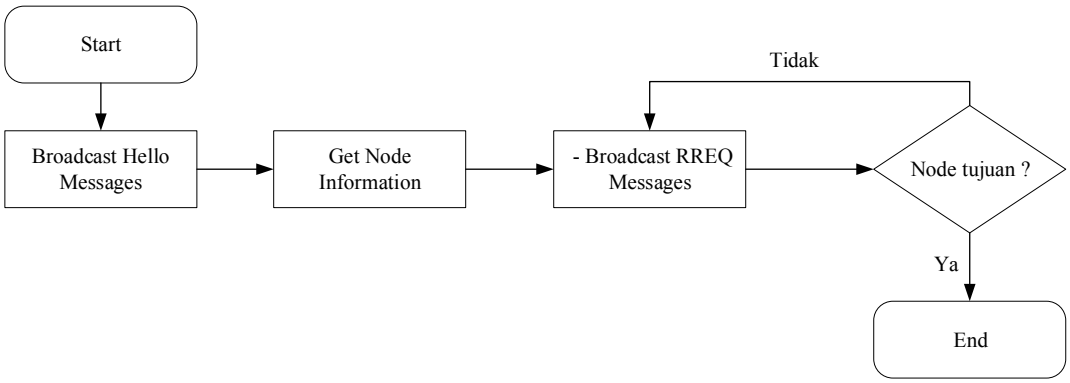


Gambar 3. 2 Desain Model Sistem

Dalam Penelitian ini skenario yang digunakan agar *route discovery* dan *overlay network* dapat berjalan dengan baik maka digunakan peta berbentuk *grid* dan *real* dengan menggunakan peta di area Kota Surabaya. Peta berbentuk *grid* dan *real* yang dibuat menggunakan *tools* dari SUMO. Hasil simulasi dari SUMO yang berupa pergerakan *node-node* akan digunakan pada simulator NS-2. Protokol pengiriman data yang digunakan untuk melakukan simulasi VANET adalah GPSR dan GPSR yang telah dimodifikasi. Hasil simulasi VANETs yang berupa *trace file* kemudian dianalisis menggunakan skrip AWK untuk mendapatkan *packet delivery ratio*, *end to end delay, routing overhead* dan rute pengiriman paket. Hasil analisis ini digunakan untuk mengukur tingkat reliabilitas pengiriman data antara protokol GPSR dengan GPSR modifikasi.

Untuk mekanisme *route discovery* pada penelitian ini mengadopsi *routing Dynamic Source Routing* (DSR) dalam menentukan rute yang menyediakan *availability* tinggi. Secara umum *routing* GPSR menentukan *relay* nodenya dengan memilih *node* yang paling terdekat dengan *node* tujuan, dengan menggunakan *route discovery* DSR proses pemilihan *node relay* menjadi lebih *reliable* karena sebelum memilih *node relay*, *node* pengirim harus mengirimkan *broadcast* RREQ menuju *node* tujuan yang kemudian akan dibalas dengan *message* RREP oleh *node* tujuan sehingga memastikan ketersediaan *relay node* lebih terjamin. Selain itu, proses ini juga akan menyimpan titik posisi *node* yang memiliki posisi sumbu x dan y yang berbeda untuk mengaplikasikan *overlay network*.

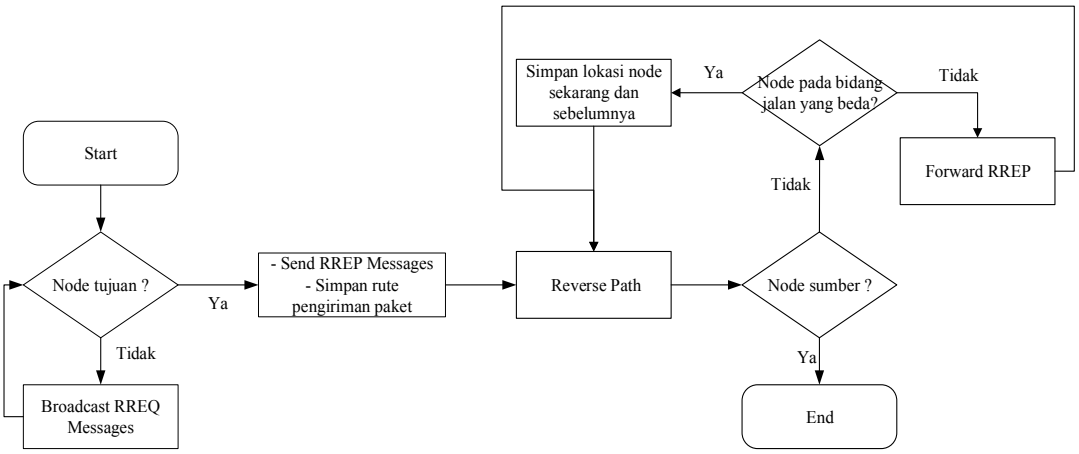
3.2.1 Mekanisme *Route Request (*RREQ*)*



Gambar 3.3 *Flowchart route request (*RREQ*)*

Alur *Route Request (*RREQ*)* setiap *node* melakukan pertukaran *hello message* untuk mendapatkan informasi *node* tetangganya kemudian melakukan *broadcast route request (*RREQ*)* menuju semua *node* tetangga hingga menemukan *node* tujuan.

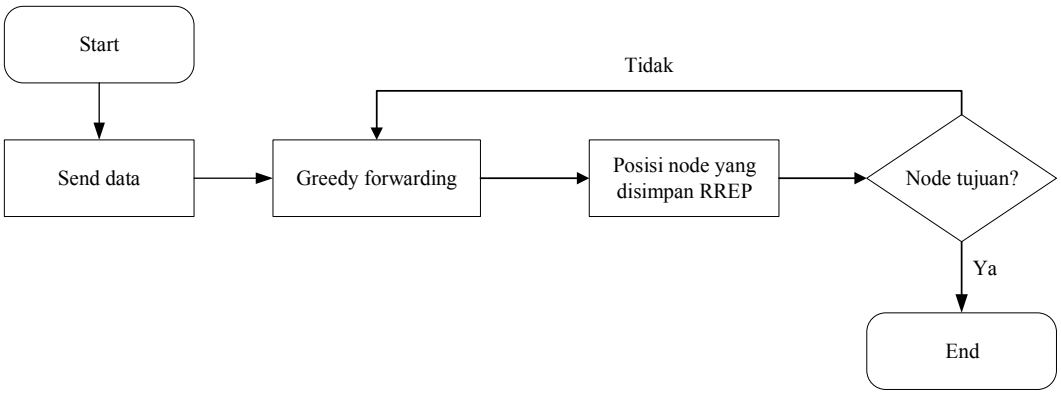
3.2.2 Mekanisme *Route Reply (*RREP*)*



Gambar 3.4 *Flowchart route reply (*RREP*)*

Pada proses ini *node* tujuan tang telah mendapat paket *route request* (RREQ) kemudian merespon dengan paket *route reply* (RREP) sebagai tanda bahwa proses komunikasi telah terbentuk. Rute yang digunakan adalah rute yang digunakan pada saat melakukan *route request* (RREQ) sehingga jalur pengiriman paket RREP seperti berjalan mundur (*drawback*). Pada saat sebuah *node* membawa paket RREP, *node* tersebut juga akan membandingkan posisi *node* sebelumnya dan *node* selanjutnya, apabila memiliki perbedaan terhadap sumbu x dan sumbu y maka posisi dari kedua *node* tersebut akan disimpan pada *header* RREP yang nantinya digunakan agar konsep *overlay* dapat diaplikasikan dan dianggap sebagai sebuah *intersection*.

3.2.3 Mekanisme *Forwarding Node*



Gambar 3.5 *Flowchart overlay network*

Ketika *route discovery* DSR telah terbentuk maka proses *forwarding* data dapat dilakukan. *Node* pengirim dapat mengirimkan data menuju *node* tujuan dengan menggunakan posisi rute yang telah ditandai secara *greedy forwarding.* Modifikasi *forwarding node* pada GPSR dilakukan dengan cara mengubah posisi tujuan pada *header* GPSR. Posisi tujuan sebelumnya merupakan posisi dari *node* tujuan. Setelah modifikasi, posisi tujuan pada *header* GPSR merupakan posisi yang harus dilalui paket data yang merupakan hasil dari RREP dan disimpan oleh *node* inisiator. Posisi-posisi yang disimpan *node* inisiator merupakan implemesntasi dari konsep *overlay network* yang telah dibahas sebelumnya. Jika posisi pertama berhasil ditempuh paket data yang ditandai dengan paket data tersebut berhasil menemukan *node* disekitar posisi pertama, maka posisi tujuan akan diubah menjadi posisi kedua, begitu seterusnya hingga paket data sampai *node* tujuan.

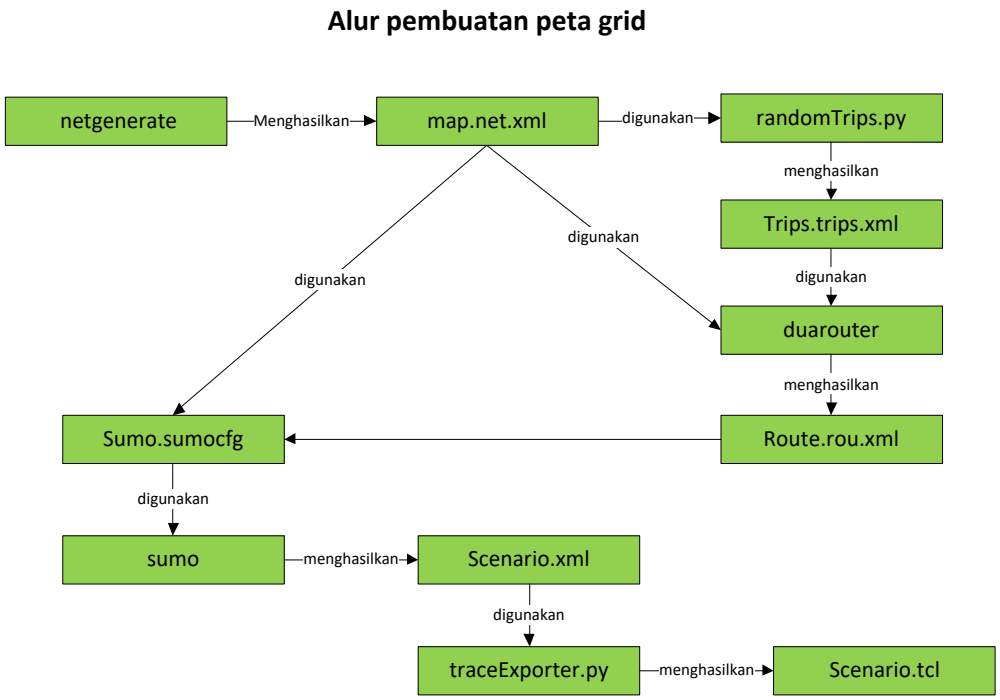
3.3 Implementasi

Pada tahap ini, dilakukan implementasi terhadap desain model sistem yang telah dirancang sebelumnya. Implementasi dilakukan menggunakan perangkat lunak simulator NS-2.35 dengan skenario peta *grid* dan peta *real*, untuk implementasi pada setiap skenario dijelaskan pada bab berikutnya.

3.3.1 Implementasi skenario *grid*

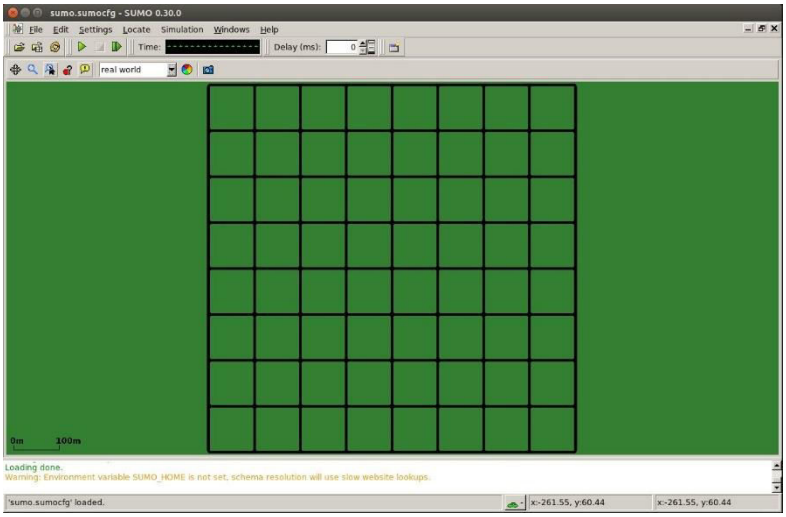
Skenario *grid* dibuat menggunakan *tools* yang telah disediakan oleh SUMO, yaitu *netgenerate.* Untuk membuat peta dengan luas 800 x 800 meter, dengan panjang antar persimpangan 100 meter dibutuhkan 9 titik persimpangan. Untuk kecepatannya menggunakan 10 m/s, 15 m/s dan 20 m/s. Sebagai contoh dengan kecepatan 10m/s. Perintah untuk membuat skenario grid adalah sebagai berikut:

```
netgenerate --grid --grid.number=9 --grid.length=100 --default.speed=10 --
tls.guess=1 --output-file=map.net.xml
```



Gambar 3.6 Alur pembuatan skneario *grid*

Tahap awal adalah membuat peta, setelah peta terbentuk tampak pada Gambar 3., proses selanjutnya adalah membuat titik awal dan titik tujuan kendaraan secara acak dengan menggunakan modul *randomTrips.py*. Tahapan selanjutnya adalah membuat rute yang akan dilalui kendaraan berdasarkan peta yang telah dibuat sebelumnya, kemudian membuat *file* .sumocfg, *file* ini digunakan untuk mendefinisikan lokasi *file.net.xml* dan *.trips.xml* agar dapat dijalankan menggunakan antarmuka simulator SUMO dan mendefinisikan durasi simulasi. *.sumocfg* diletakkan satu direktori yang sama dengan *.net.xml* dan .trips.xml. Agar dapat digunakan di NS-2, keluaran dari perintah sumo harus dikonversi ke format yang dapat dipahami oleh NS-2 melalui perintah *traceExporter.py*.



Gambar 3.7 Peta skenario *grid*

Gambar 3.7 merupakan hasil dari perintah *netgenerate*. Namun pada peta skenario di atas belum terdapat *node-node*, sehingga dibutuhkan *tools randomTrips.py* yang sudah disediakan juga oleh SUMO. Perintah untuk menggunakan *randomTrips.py* adlaah sebagai berikut:

```
randomTrips.py -n map.net.xml -e 100 -l --trip-
attributes="departLane=\"best\" departSpeed=\"max\"
departPos=\"random_free\"" --o trip.trips.xml
```

Perintah di atas akan membuat *node* sebanyak 100. *File* keluaran dari *randomTrips.py* nantinya akan digunakan oleh *tools* SUMO yang lain untuk membuat rute perjalanannya. *Tools* yang digunakan adalah *duarouter*. Perintah untuk menggunakan *duarouter* adalah sebagai berikut:

```
duarouter -n map.net.xml -t trip.trips.xml -o route.rou.xml --ignore-errors --
repair
```

Hasil dari *duarouter* nantinya akan digunakan SUMO untuk membuat skenario perjalanan *node*. SUMO membutuhkan *file* hasil dari *netgenerate* dan *duarouter*. Perintah untuk menggunakan SUMO adalah sebagai berikut:

```
sumo -b 0 -e 200 -n map.net.xml -r route.rou.xml --fcd-output=scenario.xml
```

Hasil dari SUMO selanjutnya akan dikonversi menjadi *file* dengan ekstensi *.tcl agar bisa digunakan oleh NS-2. *Tools* yang digunakan untuk melakukan konversi adalah *traceExporter.py*. Perintah untuk menggunakan *traceExporter.py* adalah sebagai berikut:

```
traceExporter.py --fcd-input=scenario.xml --ns2mobility-output=scenario.tcl
```

Pada protokol GPSR Ke Liu yang digunakan pada Tesis ini, semua *node* perlu didefinisikan pada detik ke-0 sebelum *node* melakukan pergerakan. Oleh karena itu perlu adanya modifikasi pada *randomTrips.py* untuk memudahkan dalam pembuatan skenario. Gambar 3.8 merupakan potongan kode yang ada dalam *randomTrips.py* sebelum di modifikasi. Bagian yang harus dimodifikasi adalah bagian *depart* yang harus mengeluarkan nilai 0. Untuk itu *randomTrips.py* perlu dimodifikasi yang hasil modifikasinya dapat dilihat pada Gambar 3.9. Cuplikan skenario *grid* pada *sumo-gui* dapat dilihat pada

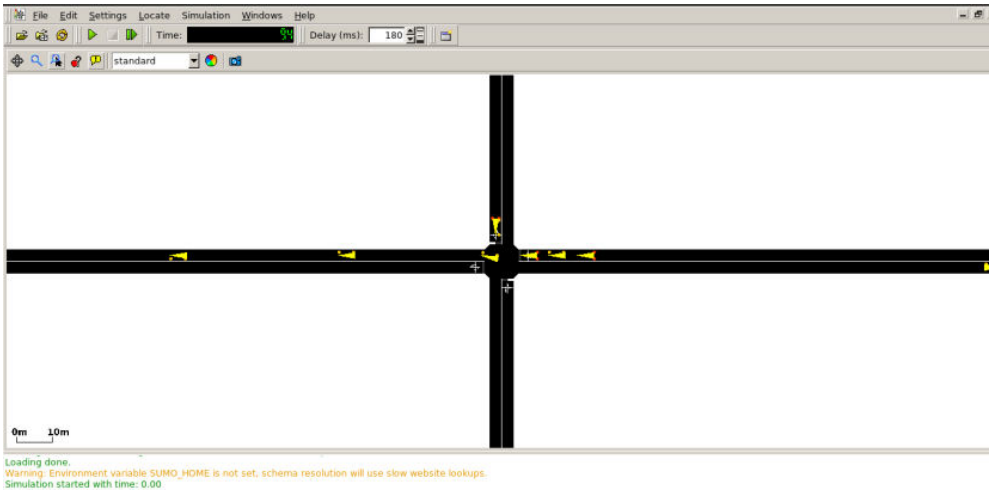
```
fouttrips.write('<trip id="%s" depart="%.2f" from="%s" to="%s"%s%s/>\n' % (
   label, depart, source_edge.getID(), sink_edge.getID(), via, options.tripattrs))
```

Gambar 3.8 *randomTrips.py* sebelum modifikasi

```
fouttrips.write('<trip id="%s" depart="%.2f" from="%s" to="%s"%s%s/>\n' % (
   label, 0, source_edge.getID(), sink_edge.getID(), via, options.tripattrs))
```

Gambar 3.9 *randomTrips.py* setelah modifikasi

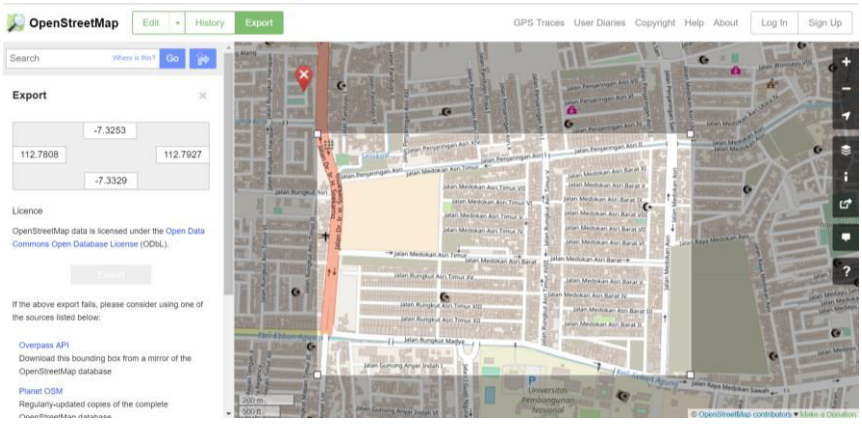
Setelah semua perintah dijalankan, visualisasi pergerakan kendaraan dapat dilihat pada Gambar 3.10 menggunakan antar muka perangkat lunak SUMO.



Gambar 3.10 Contoh pergerakan *node* pada peta *grid*

3.3.2 Implementasi skenario *real*

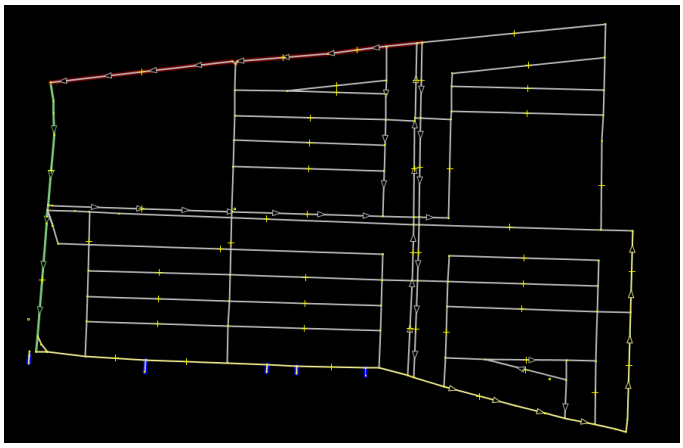
Alur pembuatan peta untuk skenario *real* hampir sama dengan pembuatan peta skenario *grid*, alur pembuatan skenario real dapat dilihat pada Gambar 3.13. Skenario *real* menggunakan peta yang diambil dari *OpenStreedMap*. Luas area yang digunakan kurang lebih sama seperti pada skenario *grid* yaitu 800 x 800 dengan kecepatan yang diseusiakan dengan lokasi peta. Peta diambil dengan cara seleksi wilayah Kota Surabaya kemudian diekspor kedalam *file* .osm melalui *browser* seperti yang ditampilkan pada Gambar 3.11.



Gambar 3.11 Peta Surabaya daerah Medokan

Peta yang diambil dari *OpenStreetMap* perlu dimodifikasi untuk menghilangkan gedung-gedung yang ada dan memperbaiki jalan-jalan yang

terputus. Aplikasi yang digunakan untuk memodifikasi adalah JOSM. Hasil modifikasi dari JOSM dapat dilihat pada Gambar 3.12.

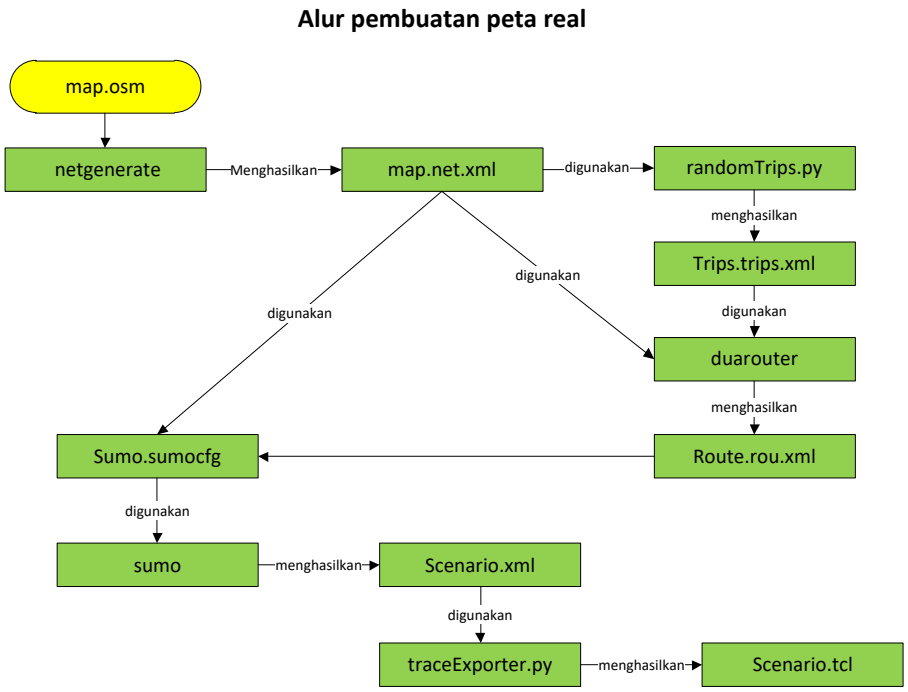


Gambar 3.12 Modifikasi menggunakan JOSM

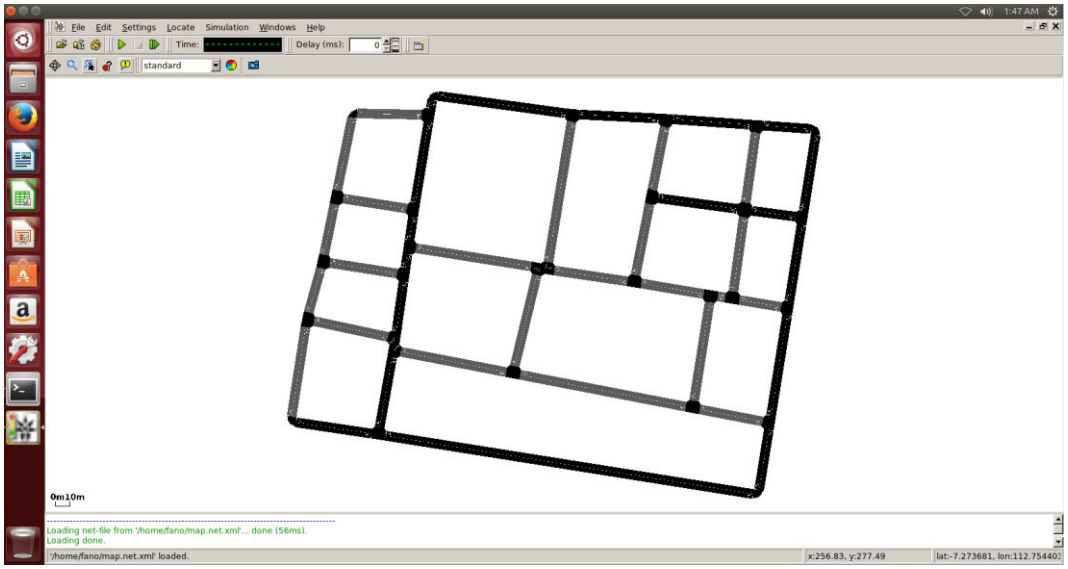
Setelah melakukan modifikasi, berarti peta sudah siap dikonversi menjadi *file* dengan ekstensi *.net.xml. *Tools* yang digunakan untuk proses konversi adalah *netconvert*. Perintah untuk melakukan proses konversi adalah sebagai berikut:

netconvert –osm-files map.osm –output-file map.net.xml

Hasil dari *netconvert* nantinya akan digunakan oleh *tools* dari SUMO yang lain untuk dijadikan bentuk *file* dengan ekstensi *.TCL. Mulai dari sini, proses pembuatan skenario *real* sama dengan proses pembuatan skenario *grid*. Cuplikan *sumo-gui* untuk skenario *real* dapat dilihat pada Gambar 3.13.



Gambar 3.13 Alur pembuatan skenario *real*



Gambar 3.14 Peta skenario *real*

Peta yang telah diekspor kemudian dapat disunting dengan perangkat lunak JOSM. Setelah penyuntingan selesai peta kemudian dikonversi menjadi *file* dengan format *.net.xml* dengan perintah *netconvert --osm-files map.osm -o map.net.xml*, hasil peta yang telah dikonversi dapat dilihat pada Gambar 3.14. Setelah peta terbentuk, langkah selanjutnya sama seperti tahapan dalam membuat

skenario *grid*, membuat titik awal dan titik tujuan kendaraan hingga konversi *file* yang dapat dijalankan dengan NS-2.

3.3.3 Implementasi Protokol

Implementasi protokol dengan cara memodifikasi protokol GPSR yang menerapkan metode *route discovery* milik protokol DSR. Modifikasi yang dilakukan adalah sebagai berikut:

- Implementasi *header* protokol DSR pada *header* GPSR
- Implementasi *routing table* pada GPSR
- Modifikasi *class* GPSRAgent
- Implementasi *route discovery* pada GPSR
- Implementasi *overlay network* pada RREP
- Modifikasi *forwarding node* pada GPSR

Data-data yang dimodifikasi untuk implementasi protokol GPSR *overlay network* adalah **gpsr.cc**, **gpsr.h**, **gpsr_packet.h**, **gpsrmod_rtable.h**, dan **gpsrmod_rtable.cc**. Data-data tersebut dapat ditemukan pada direktori **ns-2.35/gpsr**.

3.3.4 Implementasi *header* Protokol DSR pada *header* GPSR

Implementasi *header* protokol DSR pada *header* GPSR dengan cara memodifiaksi *file* **gpsr_packet.h**. Pada *file* tersebut berisi *header-header* yang digunakan oleh GPSR. Penulis membuat sebuah *struct* baru dengan nama **hdr_gpsrmod_data** yang berisi informasi RREQ, RREP, *node path, overlay node*, dan *overlay node position*.

Struct hdr_gpsrmod_data memilik satu variabel dengan tipe data *u_int8_t*, sembilan variabel dengan tipe data *integer*, dua variabel dengan tipe data *nsaddr_t*, dan satu variabel dengan tipe data *double. Node path* dan *overlay node* didefinisikan sebagai *array of nsaddr_t*, pada *node path* berisi rute yang dilalui oleh RREQ, pada *overlay node* berisi rute yang harus dilalui saat pengiriman paket yang digunakan oleh *node source*. *Overlay node position* didefinisikan sebagai *array of double* yang berisi posisi-posisi *node* yang ada pada *overlay node*. Selanjutnya perlu dilakukan

modifikasi pada **union hdr_all_gpsr** supaya *struct* hdr_gpsrmod_data dapat terbaca dengan cara menambahkan *struct* hdr_gpsrmod_data pada union hdr_all_gpsr.

```
union hdr_all_gpsr {
  hdr_gpsr          gh;
  hdr_gpsr_hello    ghh;
  hdr_gpsr_query    gqh;
  hdr_gpsr_data     gdh;
  hdr_gpsrmod_data  gpsrmod;
};
```

Gambar 3.15 Modifikasi union hdr_all_gpsr

Modifikasi union hdr_all_gpsr dapat dilihat pada Gambar 3.15. hdr_gpsrmod_data ditambahkan dalam union hdr_all_gpsr. Untuk implementasi *struct* hdr_gpsrmod_data dapat dilihat pada Gambar 3.16. Kemudian penulis melakukan pendefinisian beberapa variabel supaya hdr_gpsrmod_data dapat digunakan. **HDR_GPSRMOD_DATA(p)** digunakan untuk mengakses *struct* hdr_gpsrmod_data. **GPSRTYPE_ROUTEDISC** digunakan untuk membuat paket jenis baru pada protokol GPSR modifikasi yang dibuat. Untuk lebih jelasnya mengenai pendefinisian yang dilakukan penulis dapat dilihat pada Gambar 3.16.

```
#define GPSRTYPE_ROUTEDISC 0x03 //Route Discovery
#define HDR_GPSRMOD_DATA(p) ((struct
hdr_gpsrmod_data*)hdr_gpsr::access(p))
```

Gambar 3.16 Pendefinisian variabel pada gpsr_packet.h

```
struct hdr_gpsrmod_data {
  u_int8_t type_;
  int pValid ; //packet gpsrmod valid
  //RREQ PACKET
  int reqValid; //is RREQ
  int rq_id; //Request ID
  int rq_ttl;
  //RREP PACKET
  int repValid; //is RREP
  int rp_id; //Reply ID
  nsaddr_t nodePath[16]; //Path of node
  nsaddr_t overlayNode[16]; //list of Overlay Node;
  double path[16][2]; //loc X,Y node
  int pathLen; //length path
  int length; //length nodePath
  int index;
  inline int size(){
    int sz =
      9*sizeof(int)+
      (16*2+1)*sizeof(u_int8_t) +
      16*2*sizeof(double) ;
    return sz;
  }
};
```

Gambar 3.17 Implementasi *struct* hdr_gpsrmod_data

3.3.5 Implementasi *Routing Table* pada GPSR

Implementasi *routing table* digunakan untuk mengurangi *routing overhead* dan sebagai acuan untuk sebuah *node* apakah telah menerima paket yang sama. *File-file* yang diimplementasikan adalah **gpsrmod_rtable.h**, **gpsrmod_rtable.cc**, dan **gpsr.h**. *Routing table* bersifat lokal, maksudnya setiap *node* memiliki *routing table* sendiri sehingga *node* satu dengan yang lain memiliki *routing table* yang

berbeda. *Routing table* berisi daftar *node-node* yang melakukan RREQ. Implementasi *routing table* pada GPSR dapat dilihat pada Gambar 3.18.

```
#ifndef __gpsrmod_rtable_h__
#define __gpsrmod_rtable_h__
#include <config.h>
struct gpsrmod_rt_entry{
        nsaddr_t rt_id;
        int rt_reqno;
};
class gpsrmod_rtable{
        struct gpsrmod_rt_entry *rt;
        int len;
public:
        gpsrmod_rtable();
        int rt_lookup(nsaddr_t id);
        int rt_getrid(nsaddr_t id);
        void rt_add(nsaddr_t id, int reqno);
};
#endif
```

Gambar 3.18 Implementasi gpsrmod_rtable.h

Ketika sebuah *node* menerima paket RREQ maka *node* tersebut akan mengecek pada *routing table* yang dimilikinya. Apabila paket RREQ sudah pernah diterima *node* tersebut, maka paket RREQ tersebut akan di *drop*. Apabila paket RREQ belum pernah diterima *node* tersebut, maka *node* tersebut akan menyimpan dari mana RREQ ini dikirim dan *request id* paket RREQ tersebut. Selanjutnya menambahkan **#include gpsrmod_rtable.h** pada gpsr.h supaya *file* gpsrmod_rtable.h dan gpsrmod_rtable.cc dapat diakses dari gpsr.h.

3.3.6 Modifikasi *Class* GPSR*Agent*

Modifikasi *Class* GPSR dilakukan pada *file* **gpsr.h** dan **gpsr.cc**. Pada *file* gpsr.h, penulis melakukan pendefinisian beberapa variabel yang digunakan untuk *route discovery*. Variabel **rt** digunakan untuk mengakses *routing table,* variabel **rq_id** merupakan nomor *request* yang telah dilakukan *node* tersebut, variabel ***valid_route*** merupakan *identifier* apakah *node* tersebut sudah mempunya rute perjalanan menuju *destination*, variabel **src** dan **dst** digunakan untuk menyimpan *node source* dan *node destination*, variabel ***routeLength*** menyimpan panjang rute yang harus ditempuh paket data menuju *destination*, variabel ***thePath*** menyimpan posisi-posisi yang harus dilewati paket data menuju *destination* akhir. Untuk lebih jelasnya mengenai pendefinisian pada gpsr.h dapat dilihat pada Gambar 3.19.

```
gpsrmod_rtable *rt;
int rq_id;
bool valid_route;
nsaddr_t src;
nsaddr_t dst;
double thePath[16][2];
int routeLength;
```

Gambar 3.19 Pendefinisian variabel pada gpsr.h

Selanjutnya variabel-variabel yang telah didefinisikan seperti pada Gambar 3.19 dipanggil pada *file* gpsr.cc. Variabel-variabel tersebut diinisiasi pada *class* GPSRAgent. Gambar 3.20 menunjukkan *class* GPSRAgent pada *file* gpsr.cc beserta variabel-variabel yang ada pada Gambar 3.19.

```
GPSRAgent::GPSRAgent() : Agent(PT_GPSR),
                         hello_timer_(this), query_timer_(this),
                         my_id_(-1), my_x_(0.0), my_y_(0.0),
                         recv_counter_(0), query_counter_(0),
                         query_period_(INFINITE_DELAY)
{
  bind("planar_type_", &planar_type_);
  bind("hello_period_", &hello_period_);

  sink_list_ = new Sinks();
  nblist_ = new GPSRNeighbors();
  rt = new gpsrmod_rtable();
  src = NULL;
  dst = NULL;
  rq_id = 1;
  valid_route = 0;
  routeLength = 0;
  thePath[0][0] = NULL;

  for(int i=0; i<5; i++)
    randSend_.reset_next_substream();
}
```

Gambar 3.20 Inisiasi variabel pada *class* GPSRAgent

3.3.7 Implementasi *Route Discovery* pada GPSR

Implementasi *route discovery* pada GPSR dilakukan dengan cara menambahkan RREQ dan RREP. Untuk menambahkan RREQ dan RREP pada GPSR dapat dilakukan pada *file* gpsr.cc pada fungsi **recv()**. Kode implementasi *route discovery* pada GPSR dapat dilihat pada lampiran A.6.

3.3.8 Implementasi Konsep *Overlay Network* pada RREP

Implementasi *overlay network* pada RREP dilakukan setelah melakukan implementasi *route discovery* pada GPSR. Saat sebuah node menerima paket RREP dan paket RREP tersebut bukan untuk *node* tersebut, maka *node* tersebut akan membandingkan posisi *node* tersebut dengan *node* sebelumnya. Apabila *node* tersebut dengan *node* sebelumnya memiliki perbedaan terhadap sumbu x dan sumbu y, maka posisi dari kedua *node* tersebut akan disimpan pada *header* RREP. Kode implementasi konsep *overlay network* dapat dilihat pada lampiran A.6.

3.3.9 Modifikasi *forwarding node* pada GPSR

Modifikasi *forwarding node* pada GPSR dilakukan dengan cara mengubah posisi *destination* pada *header* GPSR. Posisi *destination* sebelumnya merupakan posisi dari *node destination.* Setelah modifikasi, posisi *destination* pada *header* GPSR merupakan posisi yang harus dilalui paket data yang merupakan hasil dari RREP dan disimpan oleh *node* inisiator. Posisi-posisi yang disimpan *node* inisiator merupakan pengimplementasian dari konsep *overlay network* yang telah dibahas sebelumnya. Jika posisi pertama berhasil ditempuh paket data yang ditandai dengan paket data tersebut berhasil menemukan *node* disekitar posisi pertama, maka posisi *destination* akan diubah menjadi posisi kedua, begitu seterusnya hingga paket data sampai node *destination*. Untuk lebih jelas mengenai modifikasi *forwarding node* pada GPSR dapat dilihat pada lampiran A.7.

3.3.10 Implementasi Simulasi pada NS-2

Implementasi simulasi pada NS-2 dilakukan pada *file-file* yang berekstensi *.tcl. Pada GPSR Ke Liu yang digunakan pada Tugas Akhir ini terdapat contoh *file* tcl untuk melakukan simulasi percobaan GPSR. Penulis menggunakan *file* yang telah diberikan patch Ke Liu untuk dijadikan *file* TCL simulasi NS-2. *File* TCL dari Ke Liu kemudian dimodifikasi sesuai kebutuhan simulasi pada Tesis ini. *File-file* TCL yang digunakan pada Tesis ini dapat dilihat pada lampiran A.1 dan lampiran A.2.

3.4 Implementasi Metrik Analisis

Penulis melakukan analisis menggunakan *file* hasil simulasi NS-2 yang berekstensi *.tr atau *tracefile*. Dari *tracefile* tersebut analisis dilakukan untuk mencari *packet delivery ratio* (PDR), *end to end delay*, dan *routing overhead*.

3.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Implementasi *packet delivery ratio* dilakukan dengan cara membandingkan data yang dikirim dengan data yang diterima. Pada Tesis ini penulis menggunakan agen CBR untuk melakukan pengiriman paket data. Untuk mendapatkan nilai PDR dari suatu *tracefile* dapat dilakukan dengan menggunakan skrip AWK yang dapat dilihat pada lampiran A.8. Perintah untuk menjalankan skirp AWK adalah sebagai berikut:

```
awk -f PDR.awk trace.tr
```

3.4.2 Implementasi *End to End Delay*

Implementasi *end to end delay* dilakukan dengan cara menghitung rata-rata *delay* antara waktu paket data dikirimkan dengan waktu paket data diterima. Pada *tracefile* yang perlu diperhatikan adalah kolom pertama, kolom kedua, kolom keempat, kolom keenam, dan kolom ketujuh. Untuk mendapatkan nilai *end to end delay* dari suatu *tracefile* dapat dilakukan dengan skrip AWK yang dapat dilihat pada lampiran A.9. Perintah untuk menjalankan skrip AWK adalah sebagai berikut:

```
awk -f e2e.awk trace.tr
```

3.4.3 Implementasi *Routing Overhead*

Implementasi *routing overhead* dilakukan dengan cara membagi antara jumlah paket *routing* dengan paket data. Paket data pada *tracefile* dapat dilihat pada baris yang memiliki tulisan cbr dan pada kolom pertama terdapat huruf r yang artinya *received*. Paket *routing* pada *tracefile* dapat dilihat pada baris yang memiliki tulisan AGT, gpsr, dan pada kolom pertamanya terdapat huruf s maupun r. Untuk mendapatkan nilai *routing overhead* dari suatu *tracefile* dapat dilakukan dengan

menggunakan skrip AWK yang dapat dilihat pada lampiran A.10. Perintah untuk menjalankan skrip AWK adalah sebagai berikut:

awk -f roverhead.awk trace.tr

3.5 Uji coba dan Simulasi

Untuk uji coba performa dari protokol maka pengujian berdasarkan beberapa metrik analisis antara lain *packet delivery ratio* (PDR), *routing overhead* dan *end to end delay* dengan variasi kecepatan kendaraan.

- PDR (*Packet delivery ratio*) adalah perbandingan antara jumlah paket yang dikirim dan diterima, formula (3.1) adalah formula untuk mencari nilai PDR.

$$PDR = \frac{Packet_{recived}}{Packet_{send}} \text{ x } 100\% \quad (3.1)$$

Keterangan:

PDR = *packet delivery ratio.*

$Packet_{recived}$ = banyak paket data yang diterima.

$Packet_{send}$ = banyak paket data yang dikirimkan.

- *Routing overhead* adalah jumlah paket *routing* yang diperlukan untuk komunikasi didalam sebuah jaringan. Rumus *routing overhead* dapat dilihat pada formula (3.2).

$$RO = \frac{routing\ packet}{data} \quad (3.2)$$

Keterangan:

RO = *routing overhead*

Routing packet = routing packet yang dibutuhkan

data = data yang berhasil dikirimkan

- *End to end delay* adalah nilai rata-rata waktu yang dibutuhkan oleh paket untuk sampai dari *source* sampai ke tujuan. Rumus *end-to-end delay* dapat dilihat pada formula (3.3).

$$E2E = \frac{\sum_{i=0}^{sent} t_{received[i]} - t_{sent[i]}}{sent} \quad (3.3)$$

Keterangan:

E2E = *end to end delay.*

$t_{received[i]}$ = waktu penerimaan paket data (detik).

$t_{sent[i]}$ = waktu pengiriman paket data (detik).

sent = banyaknya paket data yang dikirimkan.

3.6 Analisa Hasil

Pada tahap ini dilakukan analisa dari hasil uji coba dari beberapa metrik *packet delivery ratio*, *routing overhead* dan *end to end delay*. Diharapkan dari analisa uji coba ini akan diperoleh hasil yang sesuai dengan tujuan penelitian.

3.7 Penyusunan Buku Tesis

Penyusunan buku tesis dilakukan sebagai dokumentasi terhadap serangkaian penelitian yang dikerjakan agar dapat dijadikan sebagai bahan pembelajaran, referensi maupun sebuah perbaikan penelitian di masa depan yang berhubungan dengan perbaikan mekanisme *update* informasi

[halaman ini sengaja dikosongkan]

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini akan membahas uji coba dan evaluasi tentang skenario-skenario yang telah disimulasikan di NS-2.

4.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat dengan spesifikasi seperti yang dijabarkan pada Tabel 4.1.

Tabel 4.1 Spesifikasi perangkat uji coba

Komponen	Spesifikasi
CPU	Intel core i5 @1.6GHz
Sistem Operasi	Ubuntu 14.04 LTS 64-bit
Memori	4GB
Penyimpanan	100GB
Visualisasi	VMware *Workstation* 12.5.7 *build*-5813279

Sedangkan perangkat lunak yang digunakan dalam penelitian ini adalah:

1. SUMO versi 0.30.0 untuk pembuatan skenario mobilitas VANET.
2. JOSM versi 12712 untuk penyuntingan peta *OpenStreetMap*.
3. NS 2.35 untuk simulasi skenario VANET.

Uji coba dilakukan dengan menjalankan skenario-skenario yang telah dibuat dan disimulasikan pada NS-2. Hasil dari simulasi berupa *tracefile* yang akan digunakan untuk menganalisis nilai *packet delivery ratio* (PDR), *end to end delay*, dan *routing overhead* dengan bantuan skrip AWK.

4.2 Skenario Uji Coba

Skenario uji coba memiliki parameter-parameter seperti yang dijabarkan pada Tabel 4.2.

Tabel 4.2 Parameter skenario pengujian

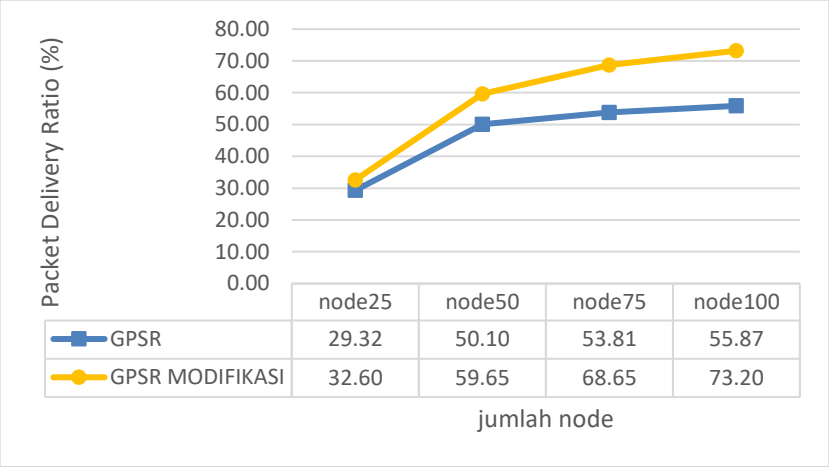
No.	Parameter	Spesifikasi
1	*Network Simulator*	NS-2.35
2	*Routing Protocol*	GPSR, GPSR modifikasi
3	Waktu Simulasi	200 detik
4	Area Simulasi	*Grid*: 800 m x 800 m *Real*: 800m x 800 m
5	Kecepatan	10 m/s, 15 m/s, 20m/s
6	Banyak *Node*	25, 50, 75, 100
7	Radius Transmisi	250m
8	*Source*/*Destination*	Tetap
9	Agen Pengirim	CBR
10	Ukuran Paket	512 *byte*
11	Protokol MAC	802.11
12	Propagasi Sinyal	*Two-ray ground*
13	Tipe Kanal	*Wireless channel*

4.3 Hasil Uji Coba Skenario *Grid*

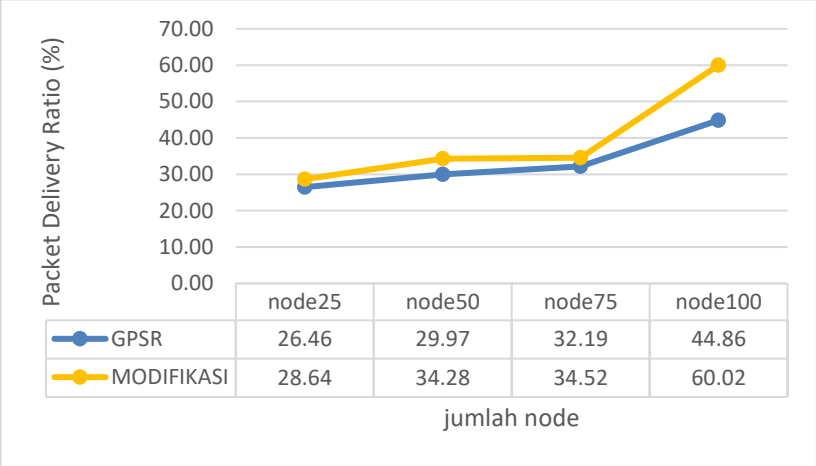
Pengujian skenario *grid* dilakukan dengan parameter-parameter yang dapat dilihat pada Tabel 4.2. Luas area simulasi adalah 800x800 meter dengan variasi *node* sebanyak 25, 50, 75, dan 100 pada variasi kecepatan 10 m/s, 15 m/s, dan 20 m/s. Untuk setiap banyak *node* pada kecepatan yang sama dilakukan pada 30 simulasi yang berbeda yang kemudian dihitung rata-rata nilainya.

4.3.1 Hasil *Packet Delivery Ratio* pada Skenario *Grid*

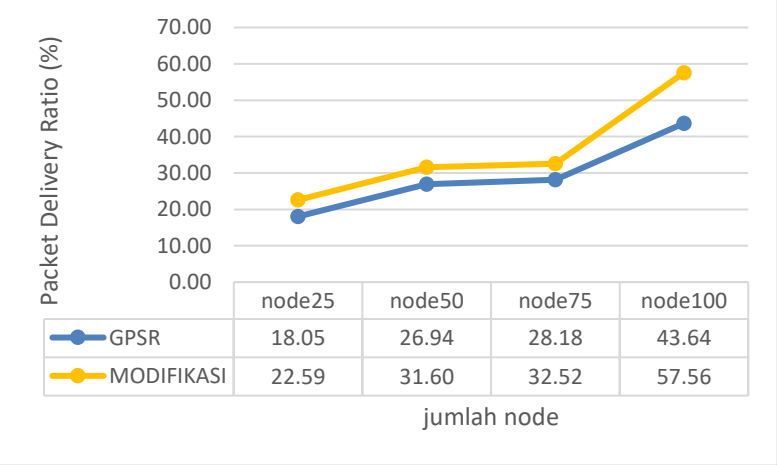
Hasil *packet delivery ratio* dengan variasi kecepatan 10 m/s, 15m/s, dan 20 m/s menunjukan GPSR modifikasi memiliki rata-rata nilai PDR lebih tinggi dibandingkan GPSR. Dengan karakteristik apabila semakin cepat kendaraan bergerak maka nilai PDR akan menurun karena pergerakan *node* mempengaruhi tingkat kesuksesan komunikasi dan semakin banyak *node* yang ada maka nilai PDR semakin tinggi.



Gambar 4.1 Grafik PDR pada kecepatan 10 m/s



Gambar 4.2 Grafik PDR pada kecepatan 15 m/s



Gambar 4.3 Grafik PDR pada kecepatan 20 m/s

Grafik dengan kecepatan 15 m/s dengan jumlah *node* 25, terdapat dua simulasi dimana protokol GPSR modifikasi memiliki nilai dibawah GPSR dikarenakan posisi *node* yang memang *random* dan kemungkinan GPSR modifikasi mendapatkan rute pengiriman paket yang jarang ditemukan *node*.

Tabel 4.3 Simulasi pada *node* 25 dengan kecepatan 15 m/s

Simulasi ke-	GPSR	GPSR modifikasi
1	83.33 %	91.66 %
2	42.55 %	44.38 %
3	29.67 %	26.78 %
4	21.5 %	23 %
5	21.11 %	19.55 %

Pada Tabel 4.3 menunjukkan bahwa tidak selalu protokol GPSR modifikasi yang unggul pada simulasi yang dilakukan. Untuk mengetahui kenapa PDR pada GPSR modifikasi lebih kecil daripada GPSR perlu dilakukan analisis pada *tracefile* milik GPSR modifikasi dan *tracefle* milik GPSR. Pada *tracefile* terlihat bahwa *node source* tidak mendapatkan *neighbor* yang lebih dekat dengan *destination* seperti yang ditunjukkan Gambar 4.4.

```
- s 20.467531392 _26_ AGT  --- 183 cbr 32 [0 0 0 0] ------- [26:0 0:0 32 0] [13] 0 0
- r 20.467531392 _26_ RTR  --- 183 cbr 32 [0 0 0 0] ------- [26:0 0:0 32 0] [13] 0 0
- s 20.467531392 _26_ RTR  --- 183 cbr 399 [0 0 0 0] ------- [26:0 0:0 31 0] [13] 0 0
```

Gambar 4.4 *Node* tidak menemukan *nexthop*

```
- s 20.467531392 _26_ AGT  --- 183 cbr 32 [0 0 0 0] ------- [26:0 0:0 32 0] [13] 0 0
- r 20.467531392 _26_ RTR  --- 183 cbr 32 [0 0 0 0] ------- [26:0 0:0 32 0] [13] 0 0
- s 20.467531392 _26_ RTR  --- 183 cbr 399 [0 0 0 0] ------- [26:0 0:0 31 0] [13] 0 0
- r 20.472425320 _0_ AGT  --- 183 cbr 399 [13a 0 1a 800] ------- [26:0 0:0 31 0]
  [13] 1 0
- s 20.489914266 _26_ RTR  --- 184 gpsr 29 [0 0 0 0] ------- [26:255 -1:255 32 0]
- r 20.490950516 _8_ RTR  --- 184 gpsr 29 [0 ffffffff 1a 800] ------- [26:255 -1:255
  32 0]
- r 20.490950587 _23_ RTR  --- 184 gpsr 29 [0 ffffffff 1a 800] ------- [26:255 -1:255
  32 0]
```

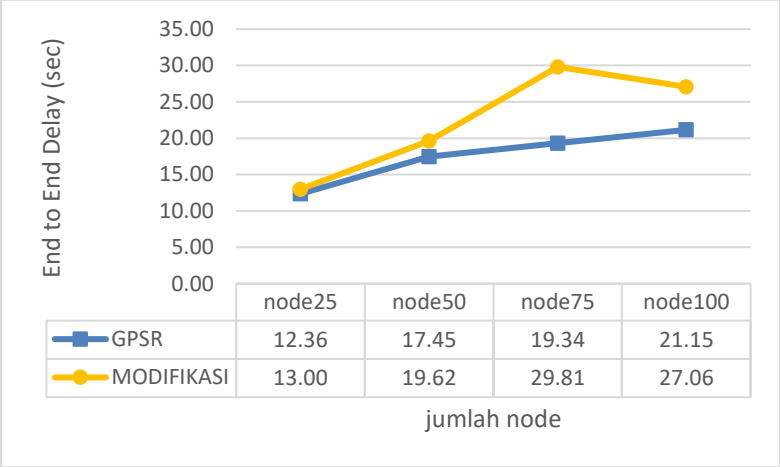
Gambar 4.5 *Node* menemukan *nexthop*

Pada Gambar 4.4, *node* 26 menginisiasi pengiriman paket data pada detik 20.20.467531392 yang ternyata tidak ditemukan *neighbor*. Gambar 4.5 menunjukkan bahwa pada detik ke-20.472425320 *node* 26 baru menemukan *nexthop* untuk *forwarding data* menuju *node destination*. Gambar 4.4 dan Gambar 4.5 merupakan salah satu contoh yang menyebabkan PDR GPSR modifikasi pada simulasi ketiga dan kelima lebih kecil daripada GPSR.

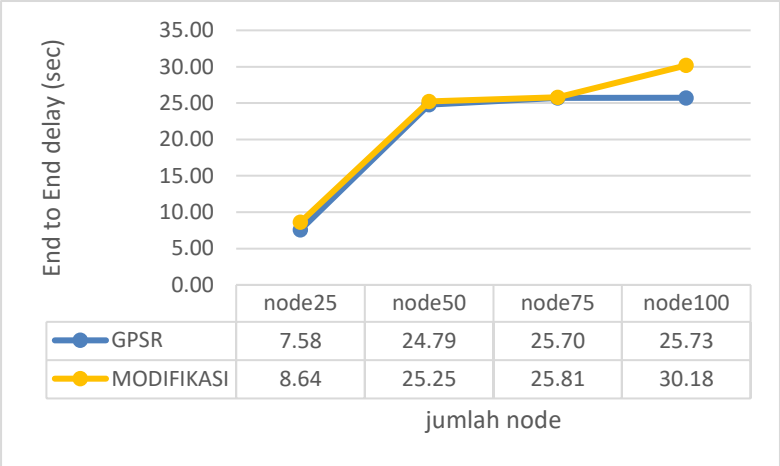
Rata-rata PDR berdasarkan Gambar 4.1, Gambar 4.2, dan Gambar 4.3 untuk protokol GPSR modifikasi adalah 32.1 %, sedangkan untuk protokol GPSR adalah 26.25 %.

4.3.2 Hasil *End to End Delay* pada Skenario *Grid*

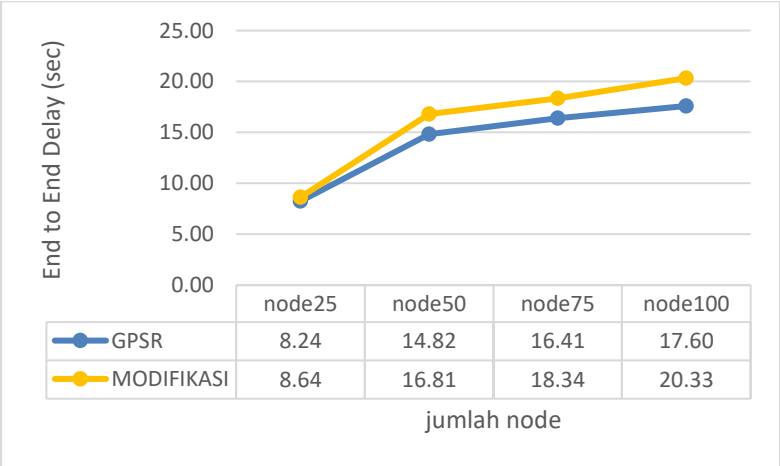
Gambar 4.6, Gambar 4.7, dan Gambar 4.8 menjelaskan grafik *end to end delay* dengan variasi *node* 25, 50, 75, dan 100 pada kecepatan 10 m/s, 15 m/s, dan 20 m/s. Pada protokol GPSR *end to end delay* untuk setiap variasi *node* rata-rata memperoleh hasil lebih baik dari pada GPSR modifikasi. Ujicoba yang dilakukan oleh penulis menggunakan skenario *full-random* pada tiap variasi node dan kecepatan sehingga pada kecepatan 10 m/s node 75 dapat mengalami peningkatan yang signifikan dikarenakan skenario yang dibuat oleh aplikasi SUMO menghasilkan node yang cenderung berjauhan sehingga menghasilkan nilai delay yang tinggi dibandingkan simulasi kecepatan lainnya.



Gambar 4.6 Grafik *end to end delay* pada kecepatan 10 m/s



Gambar 4.7 Grafik *end to end delay* pada kecepatan 15 m/s



Gambar 4.8 Grafik *end to end delay* pada kecepatan 20 m/s

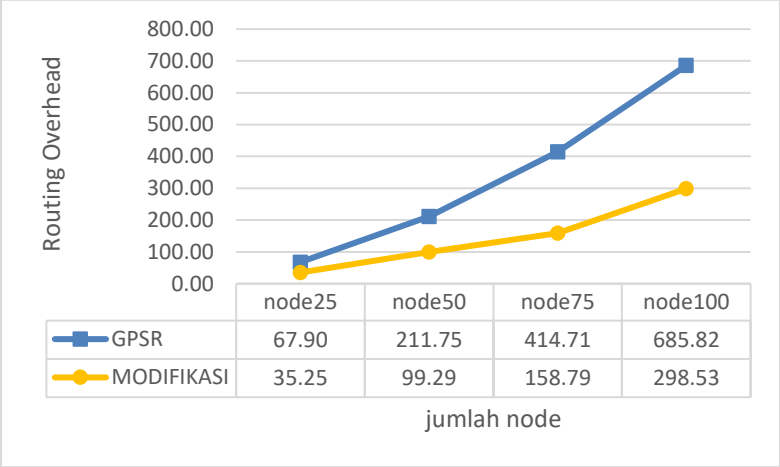
Gambar 4.7 merupakan grafik hasil *end to end delay* dengan kecepatan 15 m/s. *End to end delay* pada protokol GPSR modifikasi terlihat menunjukan nilai yang hampir sama dengan GPSR namun hasil GPSR masih lebih baik. Gambar 4.8 merupakan grafik hasil *end to end delay* dengan kecepatan 20 m/s. *End to end delay* pada protokol GPSR modifikasi mengalami kenaikan seiring bertambahnya jumlah *node* yang ada.

Dari ketiga hasil *end to end delay* pada kecepatan 10 m/s, 15 m/s, dan 20 m/s, protokol GPSR modifikasi memang lebih lama dari protokol GPSR. Hal ini dikarenakan rute perjalanan yang ada pada GPSR modifikasi memang belum tentu yang terdekat, melainkan rute tercepat yang didapat berdasarkan *route discovery* yang dilakukan. GPSR mungkin secara *end to end delay* lebih baik dari protokol GPSR modifikasi, karena GPSR benar-benar mencari *neighbor* yang terdekat dengan *destination* secara *greedy*.

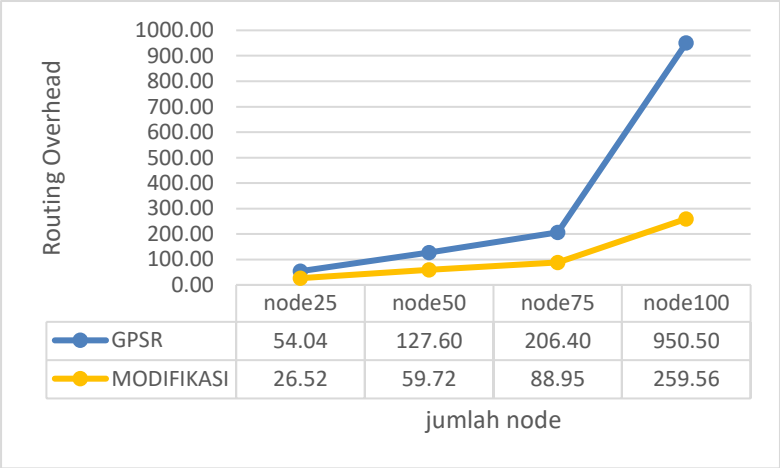
Rata-rata *end to end delay* berdasarkan Gambar 4.6, Gambar 4.7, dan Gambar 4.8 untuk protokol GPSR modifikasi adalah 19.83 ms, dan untuk protokol GPSR adalah 17 ms.

4.3.3 Hasil *Routing Overhead* pada Skenario *Grid*

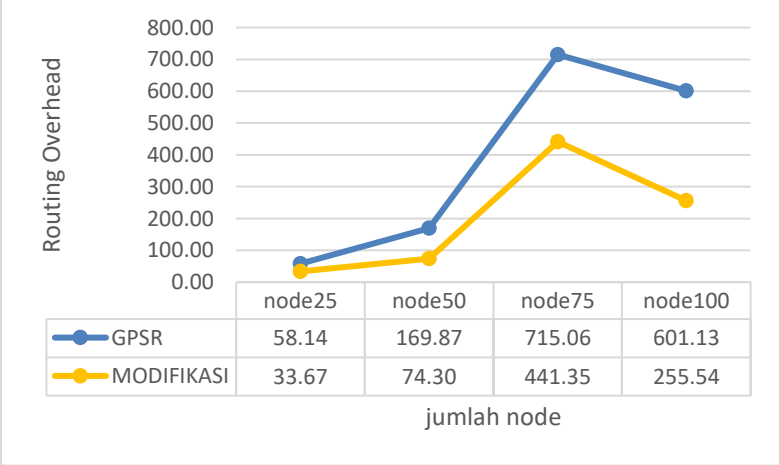
Gambar 4.9, Gambar 4.10, dan Gambar 4.11 menjelaskan grafik *routing overhead* terhadap variasi *node* 25, 50, 75, dan 100 pada variasi kecepatan 10 m/s, 15 m/s, dan 20 m/s. Pada Gambar 4.9 merupakan grafik *routing overhead* dengan kecepatan 10 m/s terlihat bahwa pada protokol GPSR modifikasi *routing overhead* mengalami kenaikan seiring bertambahnya jumlah *node* yang ada. Hal yang sama pun terjadi pada protokol GPSR. Secara keseluruhan pada grafik tersebut menunjukkan bahwa *routing overhead* pada protokol GPSR modifikasi lebih baik daripada protokol GPSR.



Gambar 4.9 Grafik *routing overhead* pada kecepatan 10 m/s



Gambar 4.10 Grafik *routing overhead* pada kecepatan 15 m/s



Gambar 4.11 Grafik *routing overhead* pada kecepatan 20 m/s

Gambar 4.10 merupakan grafik *routing overhead* pada kecepatan 15 m/s. Grafik menunjukkan bahwa baik itu protokol GPSR modifikasi maupun GPSR *routing overhead* tiap protokol mengalami kenaikan seiring bertambahnya jumlah *node* yang ada. Pada grafik ditunjukkan bahwa protokol GPSR modifikasi memiliki nilai *routing overhead* yang lebih kecil dibandingkan dengan protokol GPSR original yang memiliki nilai *routing overhead* mulai membesar seiring bertambahnya *node*.

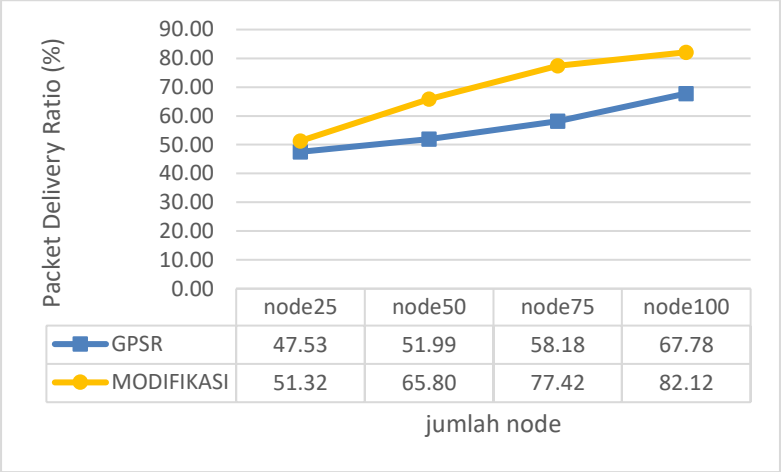
Gambar 4.11 merupakan grafik *routing overhead* pada kecepatan 20 m/s. Grafik menunjukkan bahwa pada kedua protokol baik itu GPSR modifikasi maupun GPSR, nilai *routing overhead* semakin meningkat seiring bertambahnya jumlah *node* yang ada. Pada jumlah *node* 75 menunjukan protokol GPSR modifikasi dan GPSR mengalami kenaikan yang signifikan namun GPSR modifikasi memiliki nilai yang stabil tetap dibawah *routing* protokol GPSR. Pada setiap jumlah *node* yang diuji, protokol GPSR modifikasi selalu memiliki nilai *routing overhead* yang lebih kecil dibandingkan dengan protokol GPSR.

Rata-rata *routing overhead* berdasarkan Gambar 4.9, Gambar 4.10, dan Gambar 4.11, untuk protokol GPSR modifikasi adalah 149.58 paket dan untuk protokol GPSR adalah 354.75 paket.

4.4 Hasil Uji Coba Skenario *Real*

Pengujian skenario *real* dilakukan untuk melihat performa protokol GPSR modifikasi jika diimplementasikan menggunakan peta dunia nyata yang mana bentuk jalannya tidak seperti pada skenario *grid*. Pengujian dilakukan pada luas area 800x800 meter dengan variasi node sebanyak 25, 50, 75, dan 100 pada kecepatan yang disesuaikan dengan tipe jalan pada wilayah peta yang digunakan.

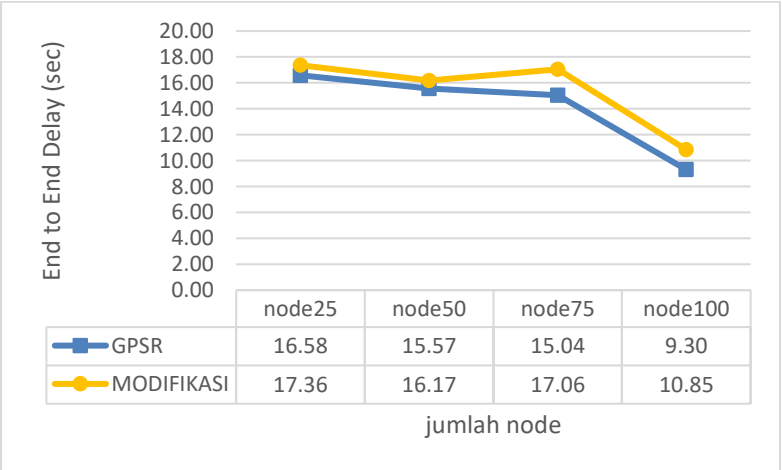
4.4.1 Hasil *Packet Delivery Ratio* pada Skenario *Real*



Gambar 4.12 Grafik *packet delivery ratio* pada skenario *real*

Gambar 4.12 merupakan grafik yang memaparkan nilai *packet delivery ratio* pada pengujian skenario *real*. Dari Grafik di atas dapat dilihat bahwa pada protokol GPSR modifikasi dan GPSR nilai *packet delivery ratio* meningkat seakan bertambahnya jumlah *node* yang ada. GPSR modifikasi mengalami peningkatan *packet delivery ratio* dari 51.32 % pada node 25 menjadi 82.12 % pada *node* 100. Hasil pada skenario real menunjukan pada tiap variasi node GPSR modifikasi menunjukan hasil PDR yang lebih baik dibandingkan GPSR.

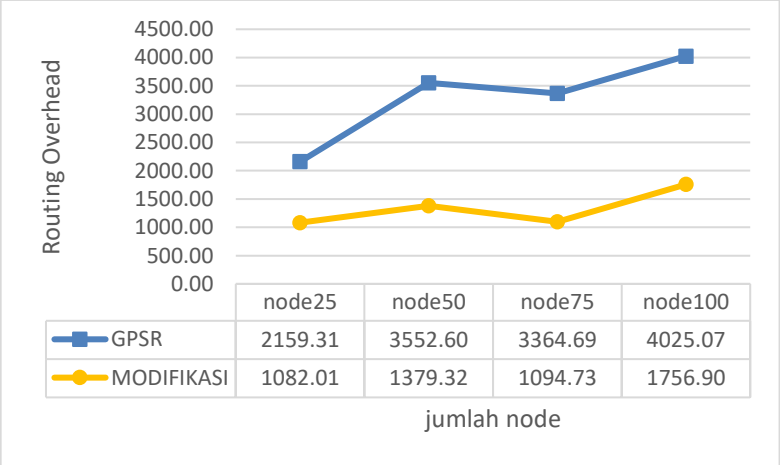
4.4.2 Hasil *End to End Delay* pada Skenario *Real*



Gambar 4.13 Grafik *end to end delay* pada skenario *real*

Gambar 4.13 merupakan grafik yang memaparkan nilai *end to end delay* pada pengujian skenario *real.* Nilai *end to end delay* protokol GPSR terlihat lebih baik dibandingkan dengan protokol GPSR modifikasi. Nilai *end to end delay* yang lebih baik belum menentukan bahwa protokol GPSR lebih baik dibandingkan dengan protokol GPSR modifikasi. Nilai *end to end delay* GPSR modifikasi lebih tinggi karena mendapatkan rute perjalanan paket data yang lebih lama dibandingkan dengan protokol GPSR yang pasti mencari yang paling dekat dengan *node destination*.

4.4.3 Hasil *Routing Overhead* pada Skenario *Real*



Gambar 4.14 Grafik *routing overhead* pada skenario *real*

Gambar 4.14 menunjukkan bahwa terjadi peningkatan nilai *routing overhead* seiring bertambahnya *node* yang ada, baik pada protokol GPSR modifikasi maupun protokol GPSR. Nilai *routing overhead* pada protokol GPSR modifikasi selalu lebih kecil pada setiap variasi *node* yang diujicobakan. Hal ini menunjukkan bahwa protokol GPSR modifikasi memiliki nilai *routing overhead* yang lebih baik dibandingkan dengan protokol GPSR.

[halaman ini sengaja dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai kesimpulan yang dapat diambil dari pengujian yang telah dilakukan dan saran yang ditujukan untuk pengembangan lebih lanjut terhadap penelitian ini.

5.1 Kesimpulan

Setelah melakukan pengujian dan evaluasi, didapatkan kesimpulan sebagai berikut:

1. *Packet Delivery Ratio* (PDR) menunjukkan bahwa pada skenario *grid* maupun skenario *real*, protokol GPSR modifikasi lebih baik dibandingkan dengan protokol GPSR. Pada skenario *grid*, rata-rata nilai PDR untuk protokol GPSR modifikasi adalah 32.1 % sedangkan rata-rata nilai PDR untuk protokol GPSR adalah 26.25 %. Pada skenario *real*, rata-rata nilai PDR untuk protokol GPSR modifikasi adalah 68.75 % sedangkan rata-rata nilai PDR untuk protokol GPSR adalah 55.75 %.
2. *End to end delay* pada protokol GPSR modifikasi lebih tinggi dibandingkan dengan protokol GPSR, hal ini dikarenakan rute perjalanan yang dihasilkan dari *route discovery* tidak selalu rute yang terdekat. Pada skenario *grid*, rata-rata nilai *end to end delay* untuk protokol GPSR modifikasi adalah 19.83 ms dan rata-rata nilai untuk protokol GPSR adalah 17 ms. Pada skenario *real*, rata-rata nilai *end to end delay* pada protokol GPSR modifikasi adalah 15 ms dan rata-rata nilai pada protokol GPSR adalah 13.75 ms.
3. *Routing Overhead* pada protokol GPSR modifikasi lebih baik daripada *routing overhead* protokol GPSR karena kesuksesan pengiriman paket data lebih baik pada protokol GPSR modifikasi. Pada skenario *grid*, rata-rata nilai *routing overhead* untuk protokol GPSR modifikasi adalah 152.08 paket, sedangkan untuk protokol GPSR adalah 354.75 paket. Pada skenario *real*, rata-rata nilai *routing overhead* untuk protokol GPSR modifikasi adalah 1.327 paket, sedangkan untuk protokol GPSR adalah 3.275 paket.

5.2 Saran

Adapun saran-saran yang diberikan untuk pengembangan sistem ini kedepannya adalah sebagai berikut:

1. Pada *node* inisiator dapat diimplementasikan rute perjalanan menuju lebih dari satu *destination*.
2. Perlu adanya metode untuk mengurangi *end-to-end delay* yang semakin tinggi seiring bertambahnya jumlah *node*.

DAFTAR PUSTAKA

Adrisano, O., Verdone, R., Nakagawa, M. (2000), "Intelligent Transportation Systems; The Role of Third Generation Mobile Radio Networks", IEEECommunications Magazine, Vol. 38, Issue. 9, pp. 144-151.

Anggoro, R., Kitasuka, T., Nakamura, R., Aritsugi, M. (2012), "Performance Evaluation of AODV and AOMDV with Probabilistic Relay in Vanet Environments", *Networking and Computing* (ICNC), pp. 259-263.

Batool, F., Khan, S. A. (2005), "Traffic Estimation And Real Time Predication Using Ad Hoc Network", *Proceedings of IEEE International Conference on Emerging Technologies*, pp. 321-329.

Biswas, S., Tatchikou, R., dan Dion, F. (2006), "Vehicle-to-vehicle Wireless Communication Protocols for Enhancing Highway Traffic Safety", *IEEE Communications Magazine*, pp. 74-82.

Dahmane, S., Lorenz, P. (2016), "Weighted Probabilistic Next-hop Forwarder Decision-Making in Vanet Environments", *Global Communications Conference* (GLOBECOM), pp. 1-6.

Djenouri, D., Soualhi, W., Nekka, E. (2008), "VANET's Mobility Models and Overtaking: An Overview", *International Confrence on Information and Communication Technologies: From Theory to Applications,* pp. 1-6.

Firdhous, M. (2011), "Multicasting Overlay Networks, A Critical Review", *International Journal of Advanced Computer Science and Applications* (IJACSA), Vol. 2, No.3, pp. 54-61.

Galan, J., Gazo-Cervero, A. (2011), "Overview and Challenges of Overlay Network: A Survey", *International Journal of Computer Science & Engineering Survey* (IJCSES), Vol. 2, No.1, pp.19-37.

Houssaini, Z. S., Zaimi, I., Oumsis, M., Ouatik, A. E. (2016), "Improvement of GPSR Protocol by Using Future Position Estimation Of Participating Nodes In Vehicular Ad-Hoc Networks", *Wireless and Mobile Communications* (WINCOM), pp. 87-94.

Hussein, S., Krings, A., Azadmanesh, A. (2017), “VANET Clock Synchronization for Ressilient DSRC Safety Applications”, *Resilience Week (RWS),* pp. 57-63.

Hu, L., Ding, Z., Shi, H. (2012), "An Improved GPSR Routing STrategy in VANET", *Wireless Communications, Networking and Mobile Computing* (WiCOM)*,* pp. 1-4.

Karp, B., Kung, H. T. (2000), “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks”, *International Conference on Mobile Computing and Networking* (MobiCom 2000), pp. 243-254.

Kim, J., Tsudik, G. (2005), “SRDP: securing route discovery in DSR”, *Mobile and Ubiquitos Systems: Networking and Services*. pp. 247-258.

Kumar, R., Routray, S. K. (2017), “Ant Colony based Dynamic Source Routing for VANET”, *Applied and Theoretical Computing and Communication Technology* (iCTAccT), pp. 279-282.

Lee, K. C., Haerri, J., Lee, U., Gerla, M. (2007), “Enhanced Perimeter Routing for Geographic Forwarding Protocols in Urban Vehicular Scenarios”, *Globecom Workshops*, pp. 1-10.

Patel, N. (2014), "A Survey Paper on Dynamic Source Routing Protocol (DSR) in Ad-Hoc Network", *International Journal for scientific Research & Development - IJSRD,* vol. 2, no.10, pp. 43-45.

Rao, S. R., Pai, M., Boussedjra, M., Mouzna, J. (2008), “GPSR-L: Greedy Perimeter Stateless Routing with lifetime for VANETs”, *ITS Telecommunications,* pp. 299-304.

Rehman, S. U., Khan, M. A., Zia, T., Zheng, L. (2013), "Vehicular ad-Hoc networks (VANETs)—An overview and challenges", *EURASIP Journal on Wireless Communications and Networking 3,* pp. 29-38.

Shelly, S., Babu, A. V. (2015), “Link Realibility based Greedy Perimeter Stateless Routing for Vehicular Ad Hoc Networks”, *International Journal of Vehicular Technology*, vol. 2015, pp. 1-16.

Shinde, P. G., Dongre, M. M. (2017), “Traffic Congestion Detection with Complex Event Processing in VNET”, *Wireless and Optical Communication Networks* (WOCN), pp. 1-5.

Silmi, N. T. (2016), "Modifikasi Protokol GPSR-MV dalam pemilihan forwarding node pada VANET", Buku Tugas Akhir, FTIF-ITS.

Zaimi, I., Houssaini, Z., Boushaba, A., Oumsis, M. (2016), “An Improved GPSR protocol to Enhance the Video Quality Transmission over Vehicular Ad hoc Networks”, *Wireless Networks and Mobile Communication* (WINCOM), pp. 146-153.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

A.1. Skenario GPSR.tcl

```
set opt(chan)       Channel/WirelessChannel
set opt(prop)       Propagation/TwoRayGround
set opt(netif)      Phy/WirelessPhy
set-
 opt(mac)           Mac/802_11
set opt(ifq)        Queue/DropTail/PriQueue    ;# for dsdv
set opt(ll)         LL
set opt(ant)        Antenna/OmniAntenna

set opt(x)          802    ;# X
set opt(y)          802    ;# Y
set opt(cp)         "./gpsrmod.tcl"
set opt(sc)         "./GpsrmodnodePos.tcl"
set opt(ifqlen)     512    ;# max packet in ifq
set opt(nn)         52            ;# number of nodes
set opt(seed)       0.0
set opt(stop)       200.0         ;# simulation time
set opt(tr)         trace.tr      ;# trace file
set opt(nam)        nam.out.tr
set opt(rp)         gpsr          ;
set opt(lm)         "off"         ;# log movement

Phy/WirelessPhy set Pt_ 0.2818    ; #250meter

# Agent/GPSR setting
Agent/GPSR set planar_type_  1   ;#1=GG planarize, 0=RNG planarize
Agent/GPSR set hello_period_   1.0 ;#Hello message period

proc usage { argv0 }  {
       puts "Usage: $argv0"
       puts "\tmandatory arguments:"
       puts "\t\t\[-x MAXX\] \[-y MAXY\]"
       puts "\toptional arguments:"
       puts "\t\t\[-cp conn pattern\] \[-sc scenario\] \[-nn nodes\]"
       puts "\t\t\[-seed seed\] \[-stop sec\] \[-tr tracefile\]\n"
}


proc getopt {argc argv} {
       global opt
       lappend optlist cp nn seed sc stop tr x y

       for {set i 0} {$i < $argc} {incr i} {
              set arg [lindex $argv $i]
              if {[string range $arg 0 0] != "-"} continue

              set name [string range $arg 1 end]
              set opt($name) [lindex $argv [expr $i+1]]
       }
}

proc log-movement {} {
    global logtimer ns_ ns

    set ns $ns_
    source ../tcl/mobility/timer.tcl
    Class LogTimer -superclass Timer
    LogTimer instproc timeout {} {
       global opt node_;
```

```
       for {set i 0} {$i < $opt(nn)} {incr i} {
           $node_($i) log-movement
       }
       $self sched 0.1
    }

    set logtimer [new LogTimer]
    $logtimer sched 0.1
}

source ../tcl/lib/ns-bsnode.tcl
source ../tcl/mobility/com.tcl

# do the get opt again incase the routing protocol file added some more
# options to look for
getopt $argc $argv

if { $opt(x) == 0 || $opt(y) == 0 } {
       usage $argv0
       exit 1
}

if {$opt(seed) > 0} {
       puts "Seeding Random number generator with $opt(seed)\n"
       ns-random $opt(seed)
}

#
# Initialize Global Variables
#
set ns_              [new Simulator]
set chan      [new $opt(chan)]
set prop      [new $opt(prop)]
set topo      [new Topography]

set tracefd   [open $opt(tr) w]
$ns_ trace-all  $tracefd

set namfile [open $opt(nam) w]
$ns_ namtrace-all $namfile

$topo load_flatgrid $opt(x) $opt(y)

$prop topography $topo

#
# Create God
#
set god_ [create-god $opt(nn)]

$ns_ node-config -adhocRouting gpsr \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF \
                 -movementTrace OFF
```

```
source ./gpsr.tcl

for {set i 0} {$i < $opt(nn) } {incr i} {
    gpsr-create-mobile-node $i
}


#
# Source the Connection and Movement scripts
#
if { $opt(cp) == "" } {
       puts "*** NOTE: no connection pattern specified."
         set opt(cp) "none"
} else {
       puts "Loading connection pattern..."
       source $opt(cp)
}

#
# Tell all the nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).000000001 "$node_($i) reset";
}
$ns_ at $opt(stop).00000001 "puts \"NS EXITING...\" ; $ns_ halt"


if { $opt(sc) == "" } {
       puts "*** NOTE: no scenario file specified."
         set opt(sc) "none"
} else {
       puts "Loading scenario file..."
       source $opt(sc)
       puts "Load complete..."
}

puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp $opt(rp)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."

proc finish {} {
    global ns_ tracefd namfile
    $ns_ flush-trace
    close $tracefd
    close $namfile
    exit 0
}

$ns_ at $opt(stop) "finish"

$ns_ run
```

A.2. Skenario Pengiriman Paket Data

```
# GPSR routing agent settings
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ at 0.00002 "$ragent_($i) turnon"
    $ns_ at 3.0 "$ragent_($i) neighborlist"
#    $ns_ at 149.0 "$ragent_($i) turnoff"
#      $ns_ at 80 "$ragent_($i) turnon"
}

$ns_ at 3.0 "$ragent_(50) startSink 10.0"

set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(50) $null_(0)

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(51) $udp_(0)

set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 32
$cbr_(0) set interval_ 1.0
$cbr_(0) set random_ 1
#    $cbr_(0) set maxpkts_ 100
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 10.0 "$cbr_(0) start"
$ns_ at 195.0 "$cbr_(0) stop"
```

A.3. Kode *Routing Table*

```
#include <gpsr/gpsrmod_rtable.h>
/*
  The Routing Table
*/
gpsrmod_rtable::gpsrmod_rtable()
{
       rt = new gpsrmod_rt_entry[128];
       len = 0;
}

int gpsrmod_rtable::rt_lookup(nsaddr_t id)
{
       for(int i=0; i<len; i++){
              if(rt[i].rt_id == id){
                     return i;
              }
       }
       return len;
}

int gpsrmod_rtable::rt_getrid(nsaddr_t id){
       int param = rt_lookup(id);
       if(param >= len){
              return 0;
       }
       return rt[param].rt_reqno;
}

void gpsrmod_rtable::rt_add(nsaddr_t id, int reqno)
{
       int param = rt_lookup(id);
```

```
        if(param < len){
                rt[param].rt_reqno = reqno;
        }

        rt[len].rt_id = id;
        rt[len].rt_reqno = reqno;
        len++;
}
```

A.4. Kode *Check Request Packet*

```
bool
GPSRAgent::checkRequest(Packet *p)
{
  struct hdr_ip *iph = HDR_IP(p);
  struct hdr_gpsrmod_data *gpsrmod = HDR_GPSRMOD_DATA(p);

  if(rt->rt_getrid(iph->saddr()) >= gpsrmod->rq_id){
    return true;
  }

  if(gpsrmod->rq_ttl == 0){ //TTL Expire
    return true;
  }

  if(iph->saddr() == my_id_){//I'm the source
    return true;
  }

  for(int i=0; i < gpsrmod->length; i++){//Already on route path
    if(gpsrmod->nodePath[i] == my_id_)
      return true;
  }
  return false;//Request OK
}
```

A.5. Kode Pengiriman RREP

```
void
GPSRAgent::sendReply(Packet *p)
{
  struct hdr_ip *iph = HDR_IP(p);
  struct hdr_gpsrmod_data *gpsrmod = HDR_GPSRMOD_DATA(p);

  //Create new RREP packet
  Packet *rp = allocpkt();
  struct hdr_cmn *rcmh = HDR_CMN(rp);
  struct hdr_ip *riph = HDR_IP(rp);
  struct hdr_gpsrmod_data *rgpsrmod = HDR_GPSRMOD_DATA(rp);

  riph->saddr() = my_id_;
  riph->sport() = RT_PORT;
  riph->daddr() = iph->saddr();
  riph->dport() = RT_PORT;
  riph->ttl()   = 255;

  //GPSRMOD header
  rgpsrmod->pValid = 1;
  rgpsrmod->repValid = 1;
  rgpsrmod->rp_id = gpsrmod->rq_id;
  rgpsrmod->type_ = GPSRTYPE_ROUTEDISC;
  rgpsrmod->length = gpsrmod->length;
  rgpsrmod->pathLen = 0;
  int param = 0;
  for(int i= rgpsrmod->length; i>=0; i--){
    rgpsrmod->nodePath[param]=gpsrmod->nodePath[i];
    param++;
  }

  //insert Loc of Destination
  MobileNode             *tempNode             =             (MobileNode
*)(Node::get_node_by_address(my_id_));
  rgpsrmod->path[rgpsrmod->pathLen][0] = tempNode->X();
  rgpsrmod->path[rgpsrmod->pathLen][1] = tempNode->Y();
  rgpsrmod->overlayNode[rgpsrmod->pathLen] = my_id_;
  rgpsrmod->pathLen++;

  //rcmh->next_hop_    =    nblist_->gf_nexthop(rgpsrmod->path[rgpsrmod-
>index][0],rgpsrmod->path[rgpsrmod->index][1]);
  rgpsrmod->index = 1;
  rcmh->next_hop_   = rgpsrmod->nodePath[rgpsrmod->index];
  rcmh->last_hop_   = my_id_;
  rcmh->addr_type_  = NS_AF_INET;
  rcmh->ptype()     = PT_GPSR;
  rcmh->size()      = IP_HDR_LEN+rgpsrmod->size();
  rcmh->direction() = hdr_cmn::DOWN;
  Scheduler::instance().schedule(this, rp, 0.010);
}
```

A.6. Kode Implementasi *Route Discovery*

```
void
GPSRAgent::recv(Packet *p, Handler *h){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);
  struct hdr_gpsrmod_data *gpsrmod = HDR_GPSRMOD_DATA(p);

  if(gpsrmod->pValid == 0){ //gpsrmod header valid
    if(iph->saddr() == my_id_){ //I'm the Source
      if(valid_route == 1){ //I have valid route
        if(src != iph->saddr() && dst != iph->daddr()){ //Destination
route is not legit
          valid_route = 0; //make Route Discovery to Destination
        }
      }
      if(valid_route == 0){ //Route Discovery
        gpsrmod->pValid = 1; //gpsrmod header valid
        //header RREQ
        gpsrmod->reqValid = 1;
        gpsrmod->rq_id = rq_id++;
        gpsrmod->rq_ttl = 20;
        gpsrmod->length = 0;
        gpsrmod->index = 0;
        gpsrmod->nodePath[gpsrmod->length] = my_id_;
        gpsrmod->length++;

        gpsrmod->type_ = GPSRTYPE_ROUTEDISC; //type

        cmh->next_hop_      = IP_BROADCAST;
        cmh->last_hop_      = my_id_;
        cmh->addr_type_     = NS_AF_INET;
        cmh->ptype()        = PT_GPSR;
        cmh->size()         = IP_HDR_LEN+gpsrmod->size();
        cmh->direction()    = hdr_cmn::DOWN;
        cmh->num_forwards() = 0;

        iph->saddr()        = my_id_;
        iph->sport()        = RT_PORT;
        iph->daddr()        = iph->daddr();
        iph->dport()        = RT_PORT;
      }
      //forward data
      else{
        if(iph->saddr() == my_id_){//a packet generated by myself
          if(cmh->num_forwards() == 0){
            struct hdr_gpsr_data *gdh = HDR_GPSR_DATA(p);
            cmh->size() += IP_HDR_LEN + gdh->size() + gpsrmod->size();

            //gpsrmod data
            gpsrmod->pValid = 0;
            gpsrmod->length = routeLength;
            gpsrmod->index = 0;
            for(int i=0; i<routeLength;i++){
              for(int j=0; j<2; j++){
                gpsrmod->path[i][j] = thePath[i][j];
              }
            }

            //gpsr data
            gdh->type_ = GPSRTYPE_DATA;
            gdh->mode_ = GPSR_MODE_GF;
            gdh->sx_ = (float)my_x_;
            gdh->sy_ = (float)my_y_;
```

```
            gdh->dx_ = (float)gpsrmod->path[gpsrmod->index][0];
            gdh->dy_ = (float)gpsrmod->path[gpsrmod->index][1];
            gdh->ts_ = (float)GPSR_CURRENT;
          }
          else if(cmh->num_forwards() > 0){ //routing loop
            if(cmh->ptype() != PT_GPSR)
             drop(p, DROP_RTR_ROUTE_LOOP);
            else Packet::free(p);
            return;
          }
        }
      }
    }
  }
  else if(gpsrmod->pValid == 1) //gpsrmod header valid
  {
    if(iph->daddr() == my_id_) { //I'm the Destination
      if(gpsrmod->reqValid){ //RREQ
        gpsrmod->nodePath[gpsrmod->length] = my_id_;
        sendReply(p);
        goto done;
      }
      else if(gpsrmod->repValid){ //RREP
        if(valid_route == 1){
          goto done;
        }

        MobileNode *prevNode = (MobileNode
*)(Node::get_node_by_address(cmh->last_hop_));
        MobileNode *thisNode = (MobileNode
*)(Node::get_node_by_address(my_id_));
        if(prevNode->X() != thisNode->X() && prevNode->Y() != thisNode-
>Y()){//check intersection
          if(gpsrmod->overlayNode[gpsrmod->pathLen-1] != cmh-
>last_hop_){ //check Overlay Node list
            gpsrmod->path[gpsrmod->pathLen][0] = prevNode->X();
            gpsrmod->path[gpsrmod->pathLen][1] = prevNode->Y();
            gpsrmod->overlayNode[gpsrmod->pathLen] = cmh->last_hop_;
          }
        }
        else{
          gpsrmod->pathLen--;
        }

        int temp = 0;
        if(gpsrmod->pathLen == 0){
          thePath[gpsrmod->pathLen][0] = gpsrmod->path[gpsrmod-
>pathLen][0];
          thePath[gpsrmod->pathLen][1] = gpsrmod->path[gpsrmod-
>pathLen][1];
          temp++;
        }
        else{
          for(int i=gpsrmod->pathLen-1; i>=0; i--){
            for(int j=0; j<2; j++){
              thePath[temp][j] = gpsrmod->path[i][j];
            }
            temp++;
          }
        }

        routeLength = temp;
        src = gpsrmod->nodePath[gpsrmod->length];
        dst = gpsrmod->nodePath[0];
```

```
        valid_route =1;
        goto done;
      }
    }
    else{ //I'm not the Destination
      if(gpsrmod->reqValid){ //RREQ
        cmh->num_forwards()++;
        gpsrmod->rq_ttl--;
        if(checkRequest(p)){
          Packet::free(p);
          goto done;
        }
        MobileNode *lastNode = (MobileNode
*)(Node::get_node_by_address(cmh->last_hop_));
        nblist_->newNB(cmh->last_hop_, lastNode->X(), lastNode->Y());
        //update RREQ header
        rt->rt_add(iph->saddr(), gpsrmod->rq_id);
        gpsrmod->nodePath[gpsrmod->length] = my_id_;
        gpsrmod->length++;
        cmh->direction() = hdr_cmn::DOWN;
        cmh->addr_type() = NS_AF_INET;
        cmh->last_hop_   = my_id_;
        cmh->next_hop_   = IP_BROADCAST;
      }
      else if(gpsrmod->repValid){ //RREP
        //check for intersection
        MobileNode *prevNode = (MobileNode
*)(Node::get_node_by_address(cmh->last_hop_));
        MobileNode *thisNode = (MobileNode
*)(Node::get_node_by_address(my_id_));
        if(prevNode->X() != thisNode->X() && prevNode->Y() != thisNode-
>Y()){
          if(gpsrmod->overlayNode[gpsrmod->pathLen-1] == cmh-
>last_hop_){ //check Overlay Node list;
            gpsrmod->path[gpsrmod->pathLen][0] = thisNode->X();
            gpsrmod->path[gpsrmod->pathLen][1] = thisNode->Y();
            gpsrmod->overlayNode[gpsrmod->pathLen] = my_id_;
            gpsrmod->pathLen++;
          }
          else{
            gpsrmod->path[gpsrmod->pathLen][0] = prevNode->X();
            gpsrmod->path[gpsrmod->pathLen][1] = prevNode->Y();
            gpsrmod->overlayNode[gpsrmod->pathLen] = cmh->last_hop_;
            gpsrmod->pathLen++;
            gpsrmod->path[gpsrmod->pathLen][0] = thisNode->X();
            gpsrmod->path[gpsrmod->pathLen][1] = thisNode->Y();
            gpsrmod->overlayNode[gpsrmod->pathLen] = my_id_;
            gpsrmod->pathLen++;
          }
        }

        cmh->num_forwards()++;
        cmh->addr_type() = NS_AF_INET;
        cmh->last_hop_ = my_id_ ;
        cmh->next_hop_ = gpsrmod->nodePath[gpsrmod->index];
        gpsrmod->index++;
        cmh->direction() = hdr_cmn::DOWN;
      }
    }
  }

  if(cmh->ptype() == PT_GPSR){
    struct hdr_gpsr *gh = HDR_GPSR(p);
    switch(gh->type_){
```

```
    case GPSRTYPE_HELLO:
      recvHello(p);
      break;
    case GPSRTYPE_QUERY:
      recvQuery(p);
      break;
    case GPSRTYPE_ROUTEDISC:
      send(p, 0);
      break;
    default:
      printf("Error with gf packet type.\n");
      exit(1);
    }
  }
  else {
    iph->ttl_--;
    if(iph->ttl_ == 0){
      drop(p, DROP_RTR_TTL);
      return;
    }
    forwardData(p);
  }
  done:
   p = 0;
   return;
}
```

A.7. Kode Implementasi *Forward Data*

```
void
GPSRAgent::forwardData(Packet *p){
  struct hdr_cmn *cmh = HDR_CMN(p);
  struct hdr_ip *iph = HDR_IP(p);

  if(cmh->direction() == hdr_cmn::UP &&
     ((nsaddr_t)iph->daddr() == IP_BROADCAST ||
      iph->daddr() == my_id_)){
    sinkRecv(p);
    printf("receive\n");
    port_dmux_->recv(p, 0);
    return;
  }
  else {
    struct hdr_gpsr_data *gdh = HDR_GPSR_DATA(p);
    struct hdr_gpsrmod_data *gpsrmod = HDR_GPSRMOD_DATA(p);

    //check this node
    if(iph->saddr() != my_id_){
      if(gpsrmod->index < gpsrmod->length-1){
        gpsrmod->index++;
      }
      gdh->sx_ = my_x_;
      gdh->sx_ = my_y_;
      gdh->dx_ = gpsrmod->path[gpsrmod->index][0];
      gdh->dy_ = gpsrmod->path[gpsrmod->index][1];
    }

    double dx = gdh->dx_;
    double dy = gdh->dy_;

    nsaddr_t nexthop;
```

```
    if(gdh->mode_ == GPSR_MODE_GF){
      nexthop = nblist_->gf_nexthop(dx, dy);
      MobileNode *nextNode = (MobileNode
*)(Node::get_node_by_address(nexthop));
      if(nexthop == -1){
              nexthop = nblist_->peri_nexthop(planar_type_, -1,gdh-
>sx_, gdh->sy_,gdh->dx_, gdh->dy_);
              gdh->sx_ = my_x_;
              gdh->sy_ = my_y_;
              gdh->mode_ = GPSR_MODE_PERI;
      }
    }
    else {
      double sddis = nblist_->getdis(gdh->sx_, gdh->sy_, gdh->dx_, gdh-
>dy_);
      double mydis = nblist_->getdis(my_x_, my_y_, gdh->dx_, gdh->dy_);
      if(mydis < sddis){
              //switch back to greedy forwarding mode
              nexthop = nblist_->gf_nexthop(dx, dy);
              gdh->mode_ = GPSR_MODE_GF;

       if(nexthop == -1){
         nexthop = nblist_->peri_nexthop(planar_type_, -1, gdh->sx_,
gdh->sy_, gdh->dx_, gdh->dy_);
         gdh->sx_ = my_x_;
         gdh->sy_ = my_y_;
         gdh->mode_ = GPSR_MODE_PERI;
       }
      }
      else{ //still perimeter routing mode
              nexthop = nblist_->peri_nexthop(planar_type_, cmh-
>last_hop_, gdh->sx_, gdh->sy_, gdh->dx_, gdh->dy_);
      }
    }
    cmh->direction() = hdr_cmn::DOWN;
    cmh->addr_type() = NS_AF_INET;
    cmh->last_hop_ = my_id_;
    cmh->next_hop_ = nexthop;
    send(p, 0);
  }
}
```

A.8. Kode AWK *Packet Delivery Ratio* (PDR)

```
BEGIN {
	sendLine = 0;
	recvLine = 0;
	forwardLine = 0;
}

$0~/^s.*AGT/{
	sendLine++
}

$0~/^r.*AGT/{
	recvLine++
}

END{
	printf"Ratio:%.4f\n",(recvLine/sendLine)*100;
}
```

A.9. Kode AWK *End to End Delay*

```
BEGIN {
	seqno = -1;
	# droppedPackets = 0;
	# receivedPackets = 0;
	count = 0;
}
{
	if($4 == "AGT" && $1 == "s" && seqno < $6) {
		seqno = $6;
	}
	if($4 == "AGT" && $1 == "s") {
		start_time[$6] = $2;
	} else if(($7 == "cbr") && ($1 == "r")) {
		end_time[$6] = $2;
	} else if($1 == "D" && $7 == "cbr") {
		end_time[$6] = -1;
	}
}
END {
	for(i=0; i<=seqno; i++) {
		if(end_time[i] > 0) {
			delay[i] = end_time[i] -
start_time[i];
			count++;
		}
		else{
			delay[i] = -1;
		}
	}
}
for(i=0; i<=seqno; i++) {
	if(delay[i] > 0) {
		n_to_n_delay = n_to_n_delay +
delay[i];
	}
}
	n_to_n_delay = n_to_n_delay/count;
	print n_to_n_delay * 1000 ;
}
```

A.10. Kode AWK *Routing Overhead*

```
BEGIN{
       recvs = 0;
       routing_packets = 0;
}

{
       if(($1 == "r") && ($7 == "cbr"))
              recvs++;
       if(($1 == "s" || $1 == "r") && ($4 == "RTR")
&& ($7 == "gpsr"))
              routing_packets++;
}

END{
       printf("\nrouting        packets:        %d",
routing_packets);
       printf("\ndata = %d", recvs);
       printf("\noverhead         :         %.3f\n",
routing_packets/recvs);
}
```

A.11. Instalasi NS-2.35

1. Instalasi dependensi yang dibutuhkan

   Sebelum melakukan instalasi NS-2 perlu dilakukan instalasi dependesi yang dibutuhkan NS-2. Perintah untuk melakukan instalasi dependensi adalah sebagai berikut:

   # sudo-apt-get install build-essenstial autoconf automake

2. Unduh *file* NS-2

   Unduh *file* NS-2 pada *link* berikut:

   https://drive.google.com/file/d/0B7S255p3kFXNSGJCZ2YzUGJDVk0/view

3. Ekstraksi *file* unduhan

   Ekstrak *file* hasil unduhan dengan perintah sebagai berikut:

   # tar -xvf ns-allinone-2.35_gcc482.tar.gz

4. Tambahkan *link path*

   Supaya NS-2 dapat dijalankan dimana saja perlu dilakukan pendefinisian *link path* pada .bashrc. Buka *file* .bashrc dengan perintah sebagai berikut:

```
# sudo gedit .bashrc
```

Tambahkan *link path* pada baris paling bawah pada *file* tersebut. *Link path* yang ditambahkan pada .bashrc adalah sebagai berikut:

```
# LD_LIBRARY_PATH
OTCL_LIB=/home/ubuntu/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/ubuntu/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_L
IB:$X11_LIB:$USR_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/ubuntu/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/home/ubuntu/ns-allinone-2.35/bin:/home/ubuntu/ns-
allinone-2.35/tcl8.5.10/unix:/home/ubuntu/ns-allinone-
2.35/tk8.5.10/unix
#the above two lines beginning from xgraph and ending with unix
should come on the same line
NS=/home/ubuntu/ns-allinone-2.35/ns-2.35/
NAM=/home/ubuntu/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

Pada pendefinisian di atas, NS-2 terletak pada direktori /home/ubuntu/ apabila NS-2 terletak pada direktori yang berbeda, ganti /home/ubuntu/ dengan direktori yang benar. Proses instalasi NS-2 akan dilanjutkan setelah dilakukan *patching* protokol GPSR.

A.12. *Patching* Protokol GPSR

1. Unduh *file patch* protokol GPSR

   Unduh *patch* GPSR pada *link* berikut:

   https://drive.google.com/file/d/0B7S255p3kFXNV2ctNFctd3JsZGs/view

2. *Patching* protokol GPSR pada NS-2

   Letakkan *file patch* pada direktori NS-2. Selanjutnya lakukan *patching* dengan perintah sebagai berikut:

   ```
   # patch -p0 < gpsr-Keliu_ns235.patch
   ```

3. Modifikasi *Makefile*

   Modifikasi *file Makefile* karena adanya modifkasi protokol gpsr dengan adanya penambahan *file* gpsrmod_rtable. Lakukan pendefinisian gpsrmod_rtable.o pada *file* Makefile di bawah gpsr/gpsr.o.

   ```
   338  gpsr/gpsr.o
   339  gpsr/gpsrmod_rtable.o
   ```

4. *Install* NS-2

   Lakukan instalasi NS-2 dengan perintah sebagai berikut:

   ```
   # ./install
   ```

[halaman ini sengaja dikosongkan]

BIODATA PENULIS

Faishal Halim S dilahirkan di Kabupaten Tegal pada 3 Desember 1992. Penulis adalah anak kedua dari tiga bersaudara laki-laki. Pendidikan ditempuh penulis dimulai dari SD Negeri Kudaile 05, SMP Negeri 1 Slawi, SMA Negeri 1 Slawi. Kemudian penulis merantau melanjutkan studi di Surakarta mengambil jurusan Teknik Informatika - Univerasitas Sebelas Maret (UNS), kemudian Teknik Informatika - Politeknik Elektronika Negeri Surabaya (PENS). Berkat usaha, kerja keras dan doa restu dari semua pihak pada tahun 2015 penulis diterima pada Pasca Sarjana Magister Teknik Informatika – Institut Teknologi Sepuluh Nopember (ITS), Fakultas Teknologi Informasi dan Komunikasi (FTIK). Penulis mengambil bidang minat Net Centric Computing (NCC). Penulis dapat dihubungi melalui account email ical@its.ac.id/faishalhalims@gmail.com